МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНИЧНИЙ ІНСТИТУТ»

На правах рукопису

Сперкач Майя Олегівна

УДК 004.9'1:[519.854+681.513]](043.3)

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПЕРАТИВНО-КАЛЕНДАРНОГО ПЛАНУВАННЯ ДРІБНОСЕРІЙНОГО ВИРОБНИЦТВА ЗА КОНЦЕПЦІЄЮ «ТОЧНО В СТРОК»

05.13.06 – Інформаційні технології

Дисертація на здобуття наукового ступеня кандидата технічних наук

Науковий керівник:
Павлов Олександр Анатолійович
Доктор технічних наук, професор

Київ – 2016

ЗМІСТ

ВСТУП

Актуальність теми. Ефективне управління виробництвом в умовах мінливості його характеру вимагає застосування сучасних концепцій управління, які забезпечують швидке і ефективне реагування на зміни. Основу такого управління складають точна та вичерпна інформація про стан виробничої, фінансової діяльності та ресурси підприємства, налагоджена система бізнес-процесів і ефективний управлінський менеджмент, невід'ємною складовою якого є оперативно-календарне планування. Планування, в таких умовах виробництва, зокрема дрібносерійного, потребує опрацювання великих обсягів інформації на досить стислих відрізках часу. Тому виправданий пошук нових підходів до ефективного розв'язання задач оперативно-календарного планування, здатних забезпечити організацію взаємоузгодженої роботи всіх підрозділів підприємства при ощадливому використанні ресурсів та виконанні планових завдань точно в строк.

Теоретичні основи реалізації такого підходу закладені у працях І. Ансоффа, М. Мескона, А. Томсона та А. Стрікленда, Д. Шендела та К. Хаттена, Дж. Пірса та Р. Робінсона. Значний внесок до їх розвитку належить Ф. Абрамсу, С. Аджирісу, М. Іпатову, Г. Мінцбергу, Т. Пітерсу, М. Портеру, Г. Саймону, В.Дж. Стівенсону, А. Стрікленду, К. Хоферу, Г. Штейнеру, С. Янгу. Результати їх досліджень виявилися ефективними та лягли в основу створення систем управління, які широко застосовуються. Пристосування цих систем до умов мінливості сучасного виробництва вимагає використання різних концепцій планування та управління, однією із яких є концепція «точно в строк». Відомі моделі та методи оперативно-календарного планування не завжди дозволяють реалізувати цю концепцію у повній мірі, тому виникає необхідність у розв'язанні нових задач, які зазвичай є важкорозв'язуваними.

Теорія важкорозв'язуваних задач набула розвитку та конкретизації у працях А.Е. Воронкова, В.В. Єлісеєва, О.С. Орлова, І.Н. Пащенка, Я.Д. Плоткіна, Г.В. Семенова, В.І. Терещенка, Е.С. Шершньова. Зусиллями багатьох учених, насамперед В.М. Глушкова, В.С. Михалевича, І.В. Сергієнка, М.З. Згуровського,

Н.З. Шора, В.С. Танаєва, В.В. Шкурби, В.Л. Волковича, О.А. Павлова, Р.Л. Грема, Е.Л. Лоулера, Ж.К. Ленстра, А.І. Семенова, В.М. Португала була розроблена система базових алгоритмів розв'язання основних класів задач оперативно-календарного планування дрібносерійного виробництва. Але відомі методи не завжди дозволяють отримувати близькі до оптимальних оперативно-календарні плани виробництва за прийнятний час, до того ж виникає потреба у інструментах формування і перевірки умов розв'язання все нових класів задач.

Таким чином, постає науково-практична задача розроблення інформаційної технології оперативно-календарного планування дрібносерійного виробництва (ІТОКПДВ) за концепцією «точно в строк», яка дозволяє будувати близькі до оптимальних за часовими критеріями календарні плани, що призводить до підвищення ефективності функціонування виробничих систем. Її розв'язання вимагає досліджень, виконаних у дисертаційній роботі.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалась у відповідності з планами наукових досліджень кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут» у рамках теми №2705-ф «Теорія ПДС-алгоритмів і створення на її основі моделей і методів планування, прийняття рішень» (номер держреєстрації 0114U003432).

Мета і завдання дослідження. *Метою дисертаційної роботи* є підвищення ефективності функціонування виробничих систем за рахунок створення, впровадження та експлуатації інформаційної технології оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк», в основу якої покладені моделі та методи, які дозволяють будувати оптимальні або близькі до оптимальних за часовими критеріями календарні плани.

Для досягнення мети дослідження необхідно виконати наступні *завдання*:

1) виконати аналітичний огляд існуючих систем планування, моделей планування, методів складання календарних планів, систем оперативно-

календарного планування виробництва;

2) розробити методи розв’язання задач складання розкладу, що вирішуються при оперативно-календарному плануванні виробництва:
   - визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним строком паралельними ідентичними пристроями;
   - визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним строком паралельними пристроями різної продуктивності;
   - мінімізації максимуму відхилення від директивного строку моментів завершення завдань паралельними ідентичними пристроями;
   - максимально рівномірного завантаження паралельних пристроїв різної продуктивності;
3) дослідити ефективність розроблених методів побудови розкладів;
4) розробити інформаційну технологію оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк», з використанням запропонованих методів;
5) здійснити експериментальне дослідження розробленої інформаційної технології.

Об’єктом дослідження є оперативно-календарне планування дрібносерійного виробництва.

Предметом дослідження є інформаційна технологія оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк».

Методи дослідження. Для виконання поставлених завдань у роботі було використано методи: системного аналізу (при проектуванні інформаційної технології); теорії розкладів, дослідження операцій, теорії складності (при розробленні методів розв’язання задач складання розкладів); комп’ютерного моделювання (при експериментальному дослідженні ефективності методів розв’язання задач складання розкладів).

Наукова новизна отриманих результатів. У дисертаційній роботі отримані такі наукові результати:

1. Вперше сформульовані достатні умови оптимальності допустимого розв’язку для важкорозв’язуваної задачі складання допустимих розкладів виконання завдань паралельними пристроями різної продуктивності з метою визначення максимально пізнього моменту початку виконання завдань із спільним директивним строком.
2. Вперше сформульовані достатні умови оптимальності допустимого розв’язку для важкорозв’язуваних задач складання допустимих розкладів виконання завдань паралельними ідентичними або з різними продуктивностями пристроями з метою мінімізації максимуму відхилення моментів завершення завдань пристроями від заданого строку.
3. Вперше на основі сформульованих достатніх умов оптимальності допустимого розв’язку для виділених задач розроблено методи їх розв’язання, які для поширених на практиці виробничих задач мають статистично сталі високі показники роботи (точність, ступінь досягнення оптимуму, час роботи); на основі сформульованих достатніх умов оптимальності визначено верхню межу відхилення значення критерію отриманих розв’язків від оптимального значення критерію.
4. Набув подальшого розвитку метод розв’язання задачі визначення максимально пізнього моменту початку виконання завдань із спільним директивним строком паралельними ідентичними пристроями, який базується на основі достатніх умов оптимальності та, на відміну від існуючих методів, має статистично сталі високі показники роботи.
5. Удосконалено інформаційну технологію оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк» за рахунок застосування розроблених ефективних методів розв’язання виробничих задач, характерних для дрібносерійного

виробництва, яка, на відміну від існуючих інформаційних технологій, призвела до зменшення часу формування планів і підвищення їх якості за часовими критеріями, згідно концепції «точно в строк».

Практичне значення отриманих результатів

Усі теоретичні напрацювання (моделі та методи) доведені до алгоритмів і реалізовані у ІТОКПДВ за концепцією «точно в строк». У сучасних умовах планування роботи виробництва незаперечним є практичне значення таких алгоритмів складання розкладів виконання завдань: визначення максимально пізнього моменту початку виконання завдань із спільним директивним строком паралельними (ідентичними або з різними продуктивностями) пристроями; мінімізації максимуму відхилення від директивного строку моментів завершення завдань паралельними ідентичними пристроями; максимально рівномірного завантаження паралельними пристроями різної продуктивності. Ці алгоритми використані при розробленні інформаційної технології оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк».

Створена ІТОКПДВ за концепцією «точно в строк» частково впроваджена в науково-технічних розробках ТОВ “БІ ДЖІ ЕФ ЦЕНТРАЛ ЄУРОП”, в якій використовуються розроблені моделі, методи та підходи до оперативно-календарного планування виробництва при розробці рішень з управління проектами. Це дозволило прискорити процес розподілу завдань між виконавцями та отримати такі плани виконання проектів, які забезпечують виконання всіх запланованих завдань максимально точно в строк. Отримано акт впровадження результатів.

Розроблену ІТОКПДВ за концепцією «точно в строк», впроваджено в приватному акціонерному товаристві «Виробничо-комерційна фірма «АС»». Її використання дозволило покращити процес планування виробництва: скоротити час планування та виготовлення виробів, заощаджувати виробничі ресурси, і тим самим збільшити об’єми виробництва. Отримано акт впровадження результатів.

Результати роботи впроваджені також у навчальний процес кафедри

автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут» і використовуються для викладання курсу «Теорії розкладів». Отримано акт впровадження результатів. Інформаційна технологія рекомендована до впровадження на підприємствах з дрібносерійним характером виробництва.

Особистий внесок здобувача. Усі результати дисертаційної роботи, які виносяться на захист, отримані автором особисто. Їх основний зміст опублікований у працях [1-10]. У працях, опублікованих у співавторстві, здобувачу належать такі результати: метод розв'язання задачі складання допустимого розкладу виконання завдань паралельними ідентичними пристроями з метою визначення максимально пізнього моменту початку виконання завдань із спільним директивним строком [1, 2]; визначення достатніх умов оптимальності та методу розв'язання задачі складання допустимого розкладу виконання завдань паралельними пристроями різної продуктивності з метою визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним строком [6]; визначення достатніх умов оптимальності та методу розв'язання задачі складання розкладу виконання завдань паралельними ідентичними пристроями з метою мінімізації максимуму відхилення від директивного строку моментів завершення завдань пристроями [5, 7]; визначення достатніх умов оптимальності та методу розв'язання задачі складання розкладу виконання завдань паралельними пристроями різної продуктивності з метою мінімізації максимуму відхилення від директивного строку моментів завершення завдань пристроями [10]; інформаційна технологія (ІТ) оперативно-календарного планування дрібносерійного виробництва [4, 8, 9].

Автор прийняв участь на паритетних засадах у розробці: загальної схеми розв'язання задачі складання допустимого розкладу виконання завдань паралель-ними ідентичними пристроями з метою визначення максимально пізнього моменту початку виконання завдань із спільним директивним строком [3]; загальної схеми розв'язання задач в багаторівневій моделі планування у складних системах [8, 9].

Апробація результатів роботи. Основні положення і результати дисертаційної роботи оприлюднені й отримали позитивну оцінку на міжнародних науково-практичних конференціях, зокрема: X міжнародній науково-практичній конференції «Математичне та імітаційне моделювання систем МОДС 2015» (м. Чернігів, 2015 р.); IX міжнародній науково-практичній конференції «Математичне та імітаційне моделювання систем МОДС 2014» (м. Чернігів, 2014 р.); VIII міжнародній науково-практичній конференції «Математичне та імітаційне моделювання систем МОДС 2013» (м. Чернігів, 2013 р.); X міжнародній науково-практичній конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» (м. Херсон, 2015 р.); міжнародній науково-технічній конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем управління організаційно-технічними та технологічними комплексами» (м. Київ, 2014 р.); міжнародній науково-практичній конференції «Актуальні проблеми економіки та управління сучасної України» (м. Ужгород, 2014 р.); XXI міжнародній конференції з автоматичного управління «Автоматика-2014» (м. Київ, 2014 р.); IV Всеукраїнській заочній науково-практичній конференції «Сучасні інформаційні технології» (м. Київ, 2013 р.); I міжнародній науково-практичній конференції молодих науковців «Інформаційні технології як інноваційний шлях розвитку України у XXI столітті» (м. Ужгород, 2012 р.).

Публікації. За результатами досліджень опубліковано 22 наукові праці, в тому числі 10 статей у наукових фахових виданнях України (з них 8 статей у виданнях України, які входять до міжнародних наукометричних баз) та 12 публікацій у збірниках матеріалів наукових конференцій.

Структура дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел із 165 найменувань та шести додатків. Загальний обсяг роботи 222 сторінки, з яких 175 сторінок основного тексту, 21 сторінка використаних джерел та 26 сторінок додатків. Робота містить 39 рисунків і 11 таблиць.

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ДО УПРАВЛІННЯ ВИРОБНИЦТВОМ

Сучасний економічний розвиток виробництва характеризується швидкою зміною технологічних процесів, зростанням кількості багатономенклатурних виробництв, для яких оптимальне планування та управління є запорукою нормального функціонування та виживання в умовах гострої конкуренції, ускладненням внутрішніх та зовнішніх зв’язків. Усе це стимулює необхідність впровадження засобів автоматизації, обчислювальної техніки, економіко-математичних методів прийняття рішень і створює умови для розвитку нових концепцій виробництва, а саме інтегрованих автоматизованих виробництв, гнучких виробничих систем, а також для автоматизації управління шляхом розробки та впровадження автоматизованих систем управління виробництвом. Однак у переважній більшості випадків і досі ще застосовуються застарілі методи планування та оперативного управління виробництвом, які не відповідають сучасному рівню виробництва та управління, новим методам обробки інформації.

В умовах ринкової економіки виникає суттєва потреба принципово нової організації виробничих процесів. Це пов’язано із необхідністю швидкої реакції виробництва на потреби ринку при одночасному зростанні асортименту продукції та зниженні трудовитрат. Отже, сучасне підприємство повинно бути адаптованим до умов випуску продукції невеликими партіями з частими змінами асортименту виробів. Ця адаптація значно спрощується за умови впровадження на підприємствах автоматизованих систем управління виробництвом. Це, у свою чергу, сприяє покращенню організації рівномірної, ритмічної взаємоузгодженої роботи всіх виробничих підрозділів підприємства.

В основі автоматизованих систем планування та управління виробництвом лежить реалізація моделей достатньо складної структури, зокрема моделей теорії розкладів.

1.1 Способи організації виробництва

Під способом організації виробництва розуміється сукупність ознак, що визначають організаційно-технічну характеристику виробничого процесу, здійснюваного на одному або багатьох робочих місцях, у масштабі дільниці, цеху, підприємства.

Різні способи організації виробництва представлені на рис. 1.1 та детально описані в науковій праці [11].

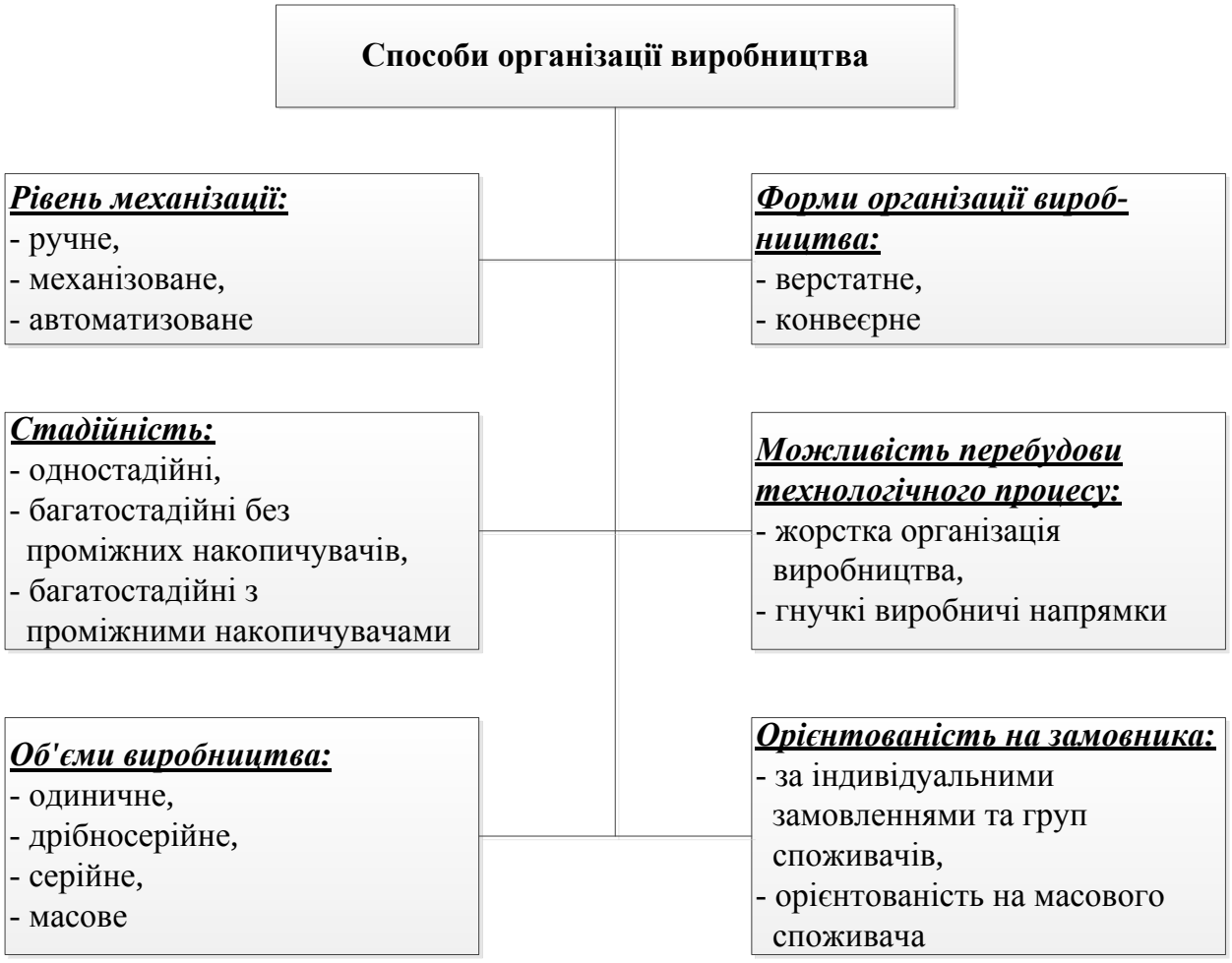


Рис. 1.1. Способи організації виробництва

Залежно від об’єму виробництва одного й того ж типу виробів розрізняють одиничні або дрібносерійні виробництва, серійні та масові або багатосерійні виробництва. У роботі буде йти мова саме про дрібносерійне виробництво, тому розглянемо його детальніше.

Дрібносерійне виробництво [12] – це виробництво малими партіями різноманітного асортименту різної продукції, що найчастіше вимагає різного

набору й послідовності технологічних операцій.

Для дрібносерійного виробництва дуже рідко всі завдання обробляються на одних і тих же пристроях та в одній і тій же послідовності, зауважимо, що це характерно і для одиничного виробництва, тому розглянемо і його.

Коефіцієнт закріплення операцій для одиничного виробництва зазвичай вище 40, а для дрібносерійного виробництва від 21 до 40 (коефіцієнт закріплення операцій для групи пристроїв визначається як відношення кількості всіх технологічних операцій, які виконані або підлягають виконанню протягом місяця, до кількості пристроїв) [13].

Варто підкреслити, що дрібносерійне виробництво найбільше відповідає виробництву невеликих виробів, які можна виготовити за допомогою декількох етапів. Іноді цей тип процесу називають переривчастою системою, тому що в процесі бере участь багато видів завдань, з частим перемиканням з одного на інше. Прикладами такого виробництва можуть служити комерційні поліграфічні фірми, компанії, що працюють у літакобудуванні, металорізальні майстерні та інші. Часто дрібносерійне виробництво призводить до багатосерійного випуску одного чи декількох подібних виробів.

Одиничний тип виробництва характеризується виготовленням широкої номенклатури виробів у одиничних кількостях, що іноді повторюється через невизначені проміжки часу. Цей тип виробництва характеризується тим, що кожному виду завдань відповідає індивідуальна структура виробничого циклу. Перехід на виробництво іншого виду продукції вимагає переналагодження устаткування, зміни організації виробничого процесу, нових технологій [13].

1.2 Сучасні методології управління виробництвом

Результативність та ефективність управління сучасним підприємством значною мірою залежать від того, яких принципів дотримуються її керівники, приймаючи управлінські рішення, і які методології (методи) вони застосовують для їх реалізації.

Важливість значення процесу управління полягає в тому, що з його допомогою керуюча система встановлює правила дій та поведінки, обов'язкові для всіх підпорядкованих їй суб'єктів, узгоджує, об'єднує, координує й регулює їх діяльність, впливає на процес прийняття та реалізації управлінських рішень, забезпечує вибір адекватних ситуацій методів управління тощо.

Принципи управління – правила, норми управлінської діяльності, відповідно до яких створюється, функціонує та розвивається система менеджменту організації. Принципи управління є узагальненням практичного позитивного управлінського досвіду і ґрунтуються на певних законах і закономірностях розвитку методології управління виробництвом [14].

Наведемо огляд існуючих методологій управління виробництвом.

У XX столітті домінувала методологія управління виробництвом, що отримала назву «Фордизм». Фордизм – модель масового виробництва стандартизованих товарів на складальних конвеєрах з використанням низькокваліфікованих працівників, зайнятих простими операціями. Один з основних постулатів фордизму: «Виробляти великі партії виробів вигідніше, ніж дрібні», міцно укорінився серед управлінців XX століття. Але випускати великі партії виробів можуть тільки гіганти, а основна маса дрібних та середніх виробництв повинна бути гнучкою, виробляючи невеликі партії виробів під запити своїх клієнтів.

З другої половини XX століття робилося безліч спроб модифікувати фордизькі моделі. Зокрема на заводах «Тойота» в 50-х роках стали ставити експерименти, адаптуючи американські концепції масового виробництва до реалій післявоєнної промисловості Японії. Усі рішення та відкриття з часом склалися в нову методологію – Ощадне виробництво (Lean Manufacturing (LM)).

У 80-х і 90-х роках з'явилась велика кількість різних методологій та парадигм з управління виробництвом, серед яких докладніше розглянемо наступні: Швидкореагуюче виробництво (Quick Response Manufacturing (QRM)) та Активне виробництво (Agile Manufacturing (AM)). Сучасні, більш гнучкі в

порівнянні з фордизмом методології, часто об’єднують терміном «постфордизм».

Найважливіші відмінності між фордизмом та постфордизмом полягають в тому, що фордизм заснований на продукті та великих обсягах виробництва, нові методології орієнтовані на клієнта та можливість випуску невеликих партій за рахунок гнучкого переналагодження обладнання. У постфордизмі робоча сила виступає носієм компетенції та джерелом розвитку, працює командою, на відміну від фордизму, де люди наймаються на окремі місця [15].

Ощадне виробництво (Lean Manufacturing, LM)

Метод застосовується для оптимізації виробництва і підвищення конкурентоспроможності. Метою методу є побудова виробництва, здатного швидко відповідати на вимоги споживачів і отримувати прибуток при будь-якій зміні ринку, у тому числі при падінні попиту. Створення досконалої виробничої системи, яка б при надходженні замовлення миттєво поставляла необхідну продукцію, і при цьому не відбувалося накопичення проміжних запасів.

Ощадне виробництво – це системний підхід до виявлення втрат і пошуку шляхів їх усунення, для того щоб зменшити час між замовленням клієнта та відвантаженням товару; бізнес-процеси, які вимагають менше людських ресурсів, капітальних вкладень, місця для виробництва, матеріалів і часу на всіх етапах.

Дана методологія спрямована на боротьбу з втратами у всіх їх проявах: зайві складські запаси, час простою, зайві переміщення, враховуючи при цьому зручність і безпеку виконання операцій для персоналу.

Вперше у повному обсязі основні методи та ідеї ощадного виробництва реалізували в Японії. У компанії «Тойота» була створена система, мета якої – скоротити або ліквідувати діяльність, що споживає ресурси та не додає вартість, тобто ту, за яку споживач не бажає платити.

Сьогодні ця система відома як виробнича система Toyota (Toyota Production System – TPS), принципи та інструменти якої знайшли відображення в її американському варіанті – системі ощадного виробництва (Lean Production).

Багато з її складових, були ще й і в радянському варіанті – наукової організації праці (НОП).

Ощадне виробництво – це підходи, методи, спрямовані на зменшення всіх можливих витрат і збільшення продуктивності. Змінюючи виробничу систему на базі принципів ощадливого виробництва, ми скорочуємо внутрішні втрати і при цьому вивільняються люди, приміщення, енергія.

Цей підхід, спрямований на якість відповідності продукції, що випускається, встановленим вимогам. Принцип роботи з якістю за системою TPS описується як три НІ: не бери в роботу дефектні заготовки, не роби дефектну продукцію, не передавай дефектну продукцію на наступну операцію.

Переваги методу полягають у тому, що висока організованість процесів дозволяє повністю уникнути непотрібних витрат і успішно конкурувати в умовах сучасного ринку. До недоліків методу можна віднести незалученість персоналу і складності при проведенні змін у компанії. Очікуваний результат від використання цього методу, це поставка в найкоротші строки необхідної продукції у разі надходження замовлення без накопичення проміжних запасів [16].

Сьогодні концепція LM використовується в тисячах компаній, що займаються виробництвом, і навіть, в організаціях, не пов’язаних з промисловістю взагалі. Потрібно відзначити, що Lean Manufacturing пов’язаний з багатьма методологіями, що з’явилися в кінці XX століття, зокрема з «Шість сигм» (Six Sigma), націлену на зниження варіабельності процесів і стабілізацію характеристик продукції.

Загальне управління якістю (англ. Total Quality Management, TQM) – загальноорганізаційний метод безперервного підвищення якості всіх організаційних процесів. TQM був популярний в кінці 80-х та початку 90-х, проте пізніше поступився ISO 9000, Lean Manufacturing та Six Sigma. Ці три методології містять безліч схожих інструментів і методів, а також схожу філософію [15].

Згодом, у рамках концепції ощадного виробництва було виділено безліч напрямків, кожен з яких представляє собою окремий метод: Потік одиничних виробів, Канбан, Всезагальне обслуговування обладнання (Total Productive Management (TPM)), Система 5S, Швидке переналагодження (SMED), Кайдзен, Захист від дурня, Управління персоналом, Точно в строк (Just In Time (JIT)).

Деякі методи самі претендують на статус самостійної виробничої методології або концепції, тому розглянемо деякі популярні з них, а саме концепцію «точно в строк».

Сенс роботи за концепцією «точно в строк» полягає в тому, щоб на всіх фазах виробничого циклу необхідний напівфабрикат до місця подальшої виробничої операції надходив саме тоді, коли це потрібно. Така концепція є в якійсь мірі «витягаючою», тобто такою, коли дільниці, розташовані на наступних етапах виробничого циклу, як би витягають необхідну їм продукцію з попередніх дільниць.

Головна мета виробничої системи «точно в строк» – забезпечити гнучку перебудову виробництва при зміні попиту. Така система забезпечує оперативне регулювання кількості виробленої продукції на кожній стадії виробництва.

Якщо система «точно в строк» діє на всьому підприємстві, то стають непотрібними запаси матеріалів. Вони можуть бути повністю ліквідовані, що призведе також до ліквідації складських запасів і самих приміщень. З економічної точки зору запаси матеріалів грають роль носіїв витрат, «заморожених коштів». Витрати на утримання виробничих запасів зменшуються. Це призводить до зменшення обсягів витрат на виробництво. Оборотність капіталу зростає.

Відомі різні модифікації системи «точно в строк», що застосовуються у світовій практиці. В їх основі лежить розробка, вперше застосована японською компанією «Тойота» і отримала широку популярність як система «Канбан».

Швидкореагуюче виробництво (Quick Response Manufacturing, QRM)

Швидке зростання великої кількості продукції, що пропонують виробники своїм замовникам, пов’язано зокрема з: розвитком CAD / CAM (системи

автоматизованого проектування і виробництва), що дозволяє компаніям розробляти «під клієнта», а потім виробляти продукцію без несення високих додаткових витрат; розвитком Інтернет, який дозволяє замовнику без особливих зусиль оцінювати величезну кількість продукції і робити свій вибір.

Дані тенденції розвитку дають підстави вважати, що в XXI столітті буде зростати попит на невелику за обсягом і вкрай різноманітну продукцію з такими функціями, які побажають самі замовники. На цьому ґрунті і з'явилася методологія QRM, яка була сформована американським математиком Раджан Сурі і докладно описана в його монографії, яка вийшла в світ у 1998 році.

Отже, швидкореагуюче виробництво (QRM) – стратегія, що використовується компаніями для скорочення часу виконання замовлення та яка охоплює все підприємство. Мета QRM – скоротити час виконання замовлення за рахунок усіх операцій компанії, як внутрішніх, так і зовнішніх.

Якщо ощадне виробництво та інші схожі методології, засновані на зниженні витрат, спрямовані на скорочення реального часу роботи, то QRM орієнтований на зниження всього часу виконання замовлення.

Зниження всього часу виконання замовлення потенційно дає набагато більший ефект, оскільки саме простої між реальними роботами над замовленням займають більшу частину часу. Зниження часу виконання замовлення, як правило, знижує вартість продукту, підвищує його якість і робить всю компанію більш конкурентоздатною [17].

Спільний знаменник QRM називається критичним шляхом виробництва (КШВ, Manufacturing Critical-path Time) – календарний час, відлік якого починається, коли замовник робить замовлення, що проходить по критичному шляху, і закінчується тоді, коли перший виріб даного замовлення поставлено замовнику. Ключова ідея КШВ, порівняти кількість «сірого часу», що минає на реальні операції, із загальним показником КШВ.

Основні концепції QRM полягають у тому, що бізнес побудований на основі виконання роботи «для складу» (коли, щоб швидше виконати замовлення, основна номенклатура продуктів виготовляється заздалегідь та кладеться на

склад), що через помилки планування та мінливості попиту призводить до збільшення КШВ, і в результаті це приводить до того, що компанія не може швидко реагувати на потреби клієнтів. Якщо сильно спростити – краще інвестувати у верстати і стандартно швидку реалізацію замовлень, ніж у склади. Тому був започаткований перехід від функціональних цехів до QRM-осередків. У філософії QRM-осередків можна простежити деяку аналогію з Scrum командою. Орієнтація працівників усіх підрозділів на єдину мету – зниження часових витрат, а звідси і єдині параметри оцінювання роботи для її досягнення, згуртувати команду працівників.

Активне (гнучке) виробництво (Agile Manufacturing, AM)

Сьогодні однією з головних проблем для промислових компаній стає проблема невизначеності та швидких змін в бізнес-середовищі.

AM – це стратегія управління компанією, мета якої зробити виробничу компанію більш стійкою до криз, змін попиту та інших непрогнозованих змін.

Президент корпорації Хонда, в інтерв'ю журналу Business Week, з цього приводу зазначив, що: «Ми зобов'язані стати дуже гнучкими для того, щоб швидко реагувати на непередбачуване майбутнє. Століття agility (активного виробництва) вже настало».

Для компаній, що працюють по AM, властива здатність швидко реконфігурувати трудові та матеріальні ресурси, щоб не втратити можливості заробити та уникнути неприємностей. Основною перевагою концепції AM є вміння оперативно підлаштовуватися під ситуацію, що змінюється і працювати в умовах невизначеності на ринку. AM підходить для галузей, де високий рівень невизначеності (наприклад IT, споживча електроніка та ін.).

Відбувається постійна готовність до змін і відповідь на них за допомогою сценарних стратегій. Сценарії: якнайбільше інтелектуальних ресурсів і якомога менше матеріальних; постійна група кросфункціональних фахівців; група фахівців на проектах на договірній основі, а також винесення непрофільних робіт на аутсорс; штат працівників не роздувається, а люди становлять ядро компанії та зацікавлені у справі.

Принципи, на які орієнтується АМ, що допомагають компанії залишатися стійкою при мінливості ринку: розгалужена мережа партнерських організацій і постачальників, чітка організація робіт.

Таким чином, особливу увагу, в компаніях типу АМ приділяється мінімізації втрат від можливих, несподіваних негативних змін, таких як втрата контрактів або ринку збуту вироблених виробів. Одночасно, дисциплінарна, швидко розширювана команда й розгалужена партнерська мережа створюють передумови для того, щоб швидко реагувати на несподівані зміни.

Проведемо порівняння наведених методологій. Три розглянуті методології відрізняються в першу чергу стратегічною орієнтацією. LM націлене на створення більшого за допомогою мінімальних засобів. Іншими словами, LM постійно виявляє втрати будь-якого плану та їх ліквідує. QRM націлене на єдину мету – зменшення часу циклу виробництва від отримання заявки і до здачі продукту замовнику. У АМ головною метою є вдосконалення можливостей для роботи в умовах невизначеності та мінливості ринку.

Вибір тієї чи іншої методології залежить від обсягів виробництва, а також від галузі, в якій працює компанія. Якщо виробництво серійне, то головне завдання, як правило – це мінімізація витрат, тобто – LM. Компанії, що створюють невеликі партії продукту, повинні вміти виконувати замовлення швидко, тому цікавіше орієнтуватися на QRM, а ті, хто працює з індивідуальними замовленнями – можуть вибрати АМ.

Якщо говорити про інновації, то з ними набагато простіше АМ компаніям, а LM найважче. Адже головним плюсом АМ виробництва є здатність працювати у завжди змінюваних умовах, розбираючись з індивідуальними проблемами. QRM займає середнє положення [17].

З урахуванням особливостей дрібносерійного типу виробництва та поставленої мети даної роботи розробка інформаційної технології оперативно-календарного планування дрібносерійного виробництва буде виконуватися за концепцією «точно в строк», яка відноситься до методології LM та може забезпечити гнучку перебудову виробництва при зміні попиту, також вона

забезпечує оперативне регулювання виготовлення продукції на кожній стадії виробництва, що призводить до зменшення обсягів витрат на виробництво та збільшення прибутку.

1.3 Оперативне управління основним виробництвом

Планування роботи та управління виробництвом підрозділяється на техніко-економічне планування та оперативне планування. Оперативне планування підрозділяється на оперативно-виробниче планування та оперативне планування допоміжного виробництва. До складу оперативно-виробничого планування входить оперативно-календарне планування та диспетчеризація [18].

Завданням оперативного планування є організація взаємоузгодженої роботи всіх виробничих підрозділів підприємства для забезпечення своєчасного виконання планового завдання при економному використанні ресурсів та високій якості продукції, тобто досягнення найкращих кінцевих результатів виробництва.

Завданнями оперативного планування є: розробка календарно-планових нормативів руху виробництва; складання оперативних планів і графіків для цехів, дільниць, бригад і робочих місць та їх доведення до безпосередніх виконавців; оперативний облік і контроль ходу виробництва, попередження та виявлення відхилень від передбачених планів і графіків та забезпечення стабілізації ходу виробництва.

Оперативне планування виробництва по місцю його виконання ділиться на міжцехове і внутрішньоцехове. Міжцехове планування забезпечує розробку, регулювання та контроль виконання планів всіма цехами підприємства, а також координує роботу основних, проектно-технологічних, планово-економічних та інших функціональних служб.

Змістом внутрішньоцехового планування є розробка оперативних планів і складання поточних графіків роботи виробничих дільниць, потокових ліній і окремих робочих місць на основі річних планів виробництва і продажу продукції основних цехів підприємства [19].

Сутність будь-якого управління [20] полягає у прийнятті керуючих рішень та подальшій реалізації передбачених цими рішеннями керуючих впливів на певний об’єкт управління.

Змістом оперативного управління основним виробництвом є: встановлення місця (цеху, дільниці, робочого місця) та часу (кварталу місяця, декади, зміни) виготовлення виробів, складальних одиниць, деталей; облік фактичного ходу виробничого процесу; визначення відхилень від заздалегідь встановленого плану; регулювання ходу виробництва, яке здійснюється для того, щоб ліквідувати наслідки небажаних відхилень і забезпечити своєчасне виконання основних задач оперативного управління.

Система оперативного управління основним виробництвом (ОУ ОВ) являє собою складну організаційно-планову систему, що включає: функціональну, елементну та організаційну підсистеми.

Функціональне розбиття характеризує коло функцій, які повинна виконувати система управління. Поелементне розбиття характеризує основні елементи, з яких вона складається. Організаційне розбиття характеризує систему з точки зору побудови системи управління.

У функціональному відношенні ОУ ОВ характеризується наступним чином: на рівні управління підприємством ОУ ОВ полягає в організації руху виробництва в межах року, кварталу, місяця; на рівні управління цехом такий рух здійснюється в межах кварталу, місяця, тижня (п’ятиденки); на рівні управління дільницею – в межах місяця, тижня, доби, зміни.

У поелементному відношенні в залежності від рівня ОУ ОВ змінюється по:

- використовуваному складу та кваліфікації управлінського персоналу; математичному забезпеченню задач планування виробництва;
- складу та кількості використовуваних комплексів технічних засобів;
- складу календарно-планових нормативів (КПН);
- складу та змісту планово-облікової документації;
- характером та напруженості інформаційних потоків.

В організаційному відношенні система ОУ ОВ здійснює свої функції за допомогою:

- планово-диспетчерського відділу (ПДВ) на рівні підприємства;
- планово-диспетчерського бюро (ПДБ) на цеховому рівні;
- планово-управлінського персоналу дільниці на рівні самої дільниці.

Загальними вимогами до системи ОУ ОВ є:

- наукова обґрунтованість;
- оптимальність;
- точність;
- оперативність керуючих рішень.

Наукова обґрунтованість системи ОУ ОВ припускає визначення: вибору елементів системи (планово-облікових одиниць, планово-облікового та планового періоду); вибору та розрахунку календарно-планових нормативів (періодичності і розмірів партій, тривалості виробничих циклів та випереджень запуску і випуску партій деталей та складальних одиниць, незавершеного виробництва та ін.); побудови об’ємних і оперативно-календарних планів; системи контролю та регулювання виробництва; достовірність вихідних даних.

Для оперативного планування виробництва застосовують системи планування і контролю виробництва, що сприяють організації рівномірної, ритмічної взаємоузгодженої роботи всіх виробничих підрозділів підприємства. Їх формування та застосування описані у роботах [21-29].

Система планування і контролю виробництва складається з п’яти основних рівнів [21]: розробка стратегічного бізнес-плану, формування плану виробництва (плану продажів та операцій), створення головного календарного плану виробництва, розробка плану потреби в ресурсах, формування переліку закупівель, здійснення планів та контроль над виробничою діяльністю.

У кожного рівня своє завдання, тривалість та рівень деталізації. Згідно ієрархії планування, перші чотири рівні – це рівні планування. Результатом розробки планів є ініціювання закупівлі чи виготовлення того, що необхідно.

Останній рівень – це здійснення планів за допомогою контролю виробничої діяльності та закупівель, розміщення замовлень у постачальників, та виконання їх постачальниками. На рис. 1.2 наведено взаємозв'язок усіх рівнів ієрархії планування.

Стратегічний бізнес-план – це опис головних цілей та завдань, які компанія планує виконати в строк від двох до десяти років або довше.

Виробничий план – ґрунтується на завданнях, поставлених у стратегічному бізнес-плані, управління виробничого відділу приймає рішення з таких питань: кількість виробів у кожній групі, які потрібно зробити в кожен період часу; бажаний рівень матеріально-виробничих запасів; обладнання, трудові ресурси та матеріали, необхідні в кожен період часу; наявність необхідних ресурсів.

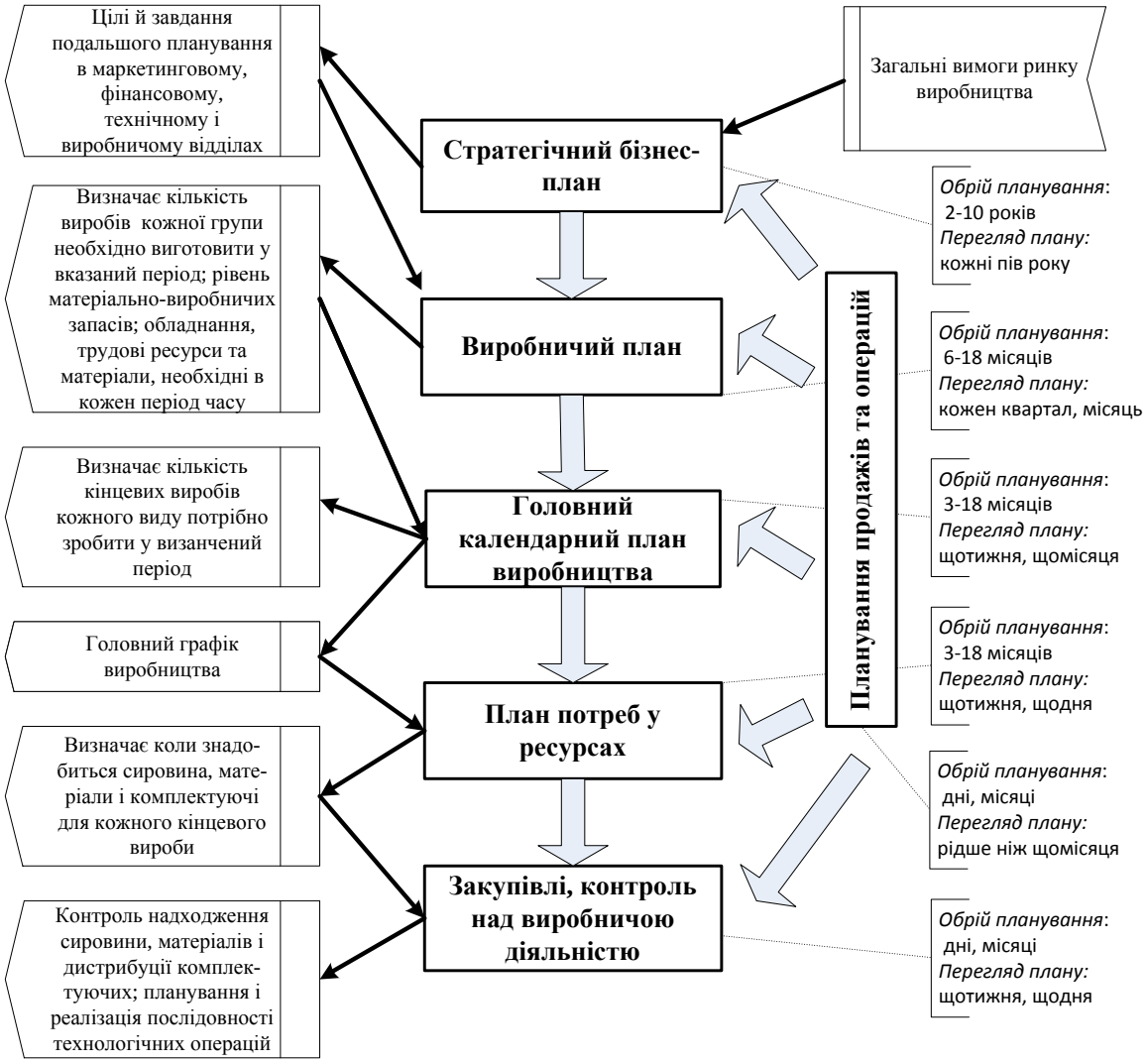


Рис. 1.2. Взаємозв'язок рівнів в ієрархії планування

Головний календарний план виробництва – це план виробництва окремих кінцевих виробів. У ньому здійснюється розбиття виробничого плану, що відображає кількість кінцевих виробів кожного виду, яке потрібно зробити в кожен період часу. Вихідною інформацією для розробки головного календарного плану виробництва є виробничий план, прогнози по окремих кінцевим виробам, замовлення на закупівлю, відомості про матеріально-виробничі запаси та існуючі продуктивності.

План потреби в ресурсах – це план виробництва і закупівлі компонентів, які використовуються при виготовленні передбачених головним календарним планом виробництва виробів. У ньому зазначені необхідні кількості та строки передбачуваного виготовлення або використання їх у виробництві.

Закупівлі та контроль над виробничою діяльністю являють собою фазу впровадження і контролю системи виробничого планування та контролю. Процес закупівель відповідає за організацію та контроль надходження сировини, матеріалів та комплектуючих на підприємство. Контроль над виробничою діяльністю – це планування послідовності технологічних операцій на підприємстві і контроль над ними.

Планування продажів та операцій – це процес, призначений для постійного перегляду стратегічного бізнес-плану та координації планів різних підрозділів.

Метою ОУ ОВ є забезпечення суворого виконання заданого плану випуску продукції за кількістю та номенклатурою і у встановлені строки на основі раціонального (оптимального) використання виробничих ресурсів, а також шляхом виявлення та мобілізації внутрішньовиробничих резервів.

Для реалізації цієї мети необхідна науково-обґрунтована побудована та функціональна система ОУ ОВ, яка має забезпечити вирішення наступних завдань:

– повне, комплектне і рівномірне виконання виробничої програми при дотриманні директивних строків випуску продукції;

– повне та найбільш доцільне (оптимальне) використання засобів виробництва і трудових ресурсів; максимальне прискорення виробництва та

забезпечення максимального використання оборотних коштів у стадії виробництва;

– забезпечення умов, що сприяють розвитку передових форм організації праці у виробництві;

– автоматизація виконання основних планово-облікових, облікових робіт та отримання документації.

До функцій системи ОУ ОВ відносяться: управління як процес прийняття рішень; планування як процес визначення лінії поведінки об’єкта управління; облік як процес контролю, аналізу та виявлення відхилень від заданої планом лінії поведінки об’єкта; регулювання як процес локалізації виникаючих відхилень і збереження заданої лінії поведінки керованого об’єкта [14].

Система ОУ ОВ включає: об’ємне планування; оперативно-календарне планування; оперативний облік; поточний контроль і регулювання.

У процесі об’ємного планування проводиться розподіл річної виробничої програми виробництва в об’ємному і натуральному вираженні між цехами та дільницями відповідно до виділених їм трудових та матеріальних ресурсів. При вирішенні завдань об’ємного планування прагнуть до забезпечення рівномірного завантаження устаткування та працівників у всіх цехах та на дільницях підприємства. Для вирішення завдань об’ємного планування використовуються методи математичного програмування.

Оперативно-календарне планування – продовження та розвиток об’ємного планування. На цьому етапі об’єктом планування є окремі вироби, складальні одиниці, деталі, деталеопераці. Воно ґрунтується на певних нормативах (розміри партій деталей, складальних одиниць та виробів; тривалість виробничих циклів, випередження запуску і випуску партій деталей; нормативи незавершеного виробництва), що дозволяють здійснити зв’язок календарних планів та узгодження роботи взаємопов’язаних робочих місць, дільниць, цехів.

Оперативно-календарне планування здійснюється як на заводському, так і на внутрішньоцеховому рівнях. У завдання заводського оперативно-календарного планування входить визначення кількості та часу передачі по

планованих позиціях (деталей, складальних одиниць, виробів) з цеху в цех по місяцях або кварталах. Частково ця робота може виконуватися на етапі об’ємного планування.

У завдання внутрішньоцехового оперативно-календарного планування входить: визначення місця і строків початку та закінчення обробки кожної деталеоперації, їх груп або деталі в цілому; складання календарного плану-графіка роботи дільниць цеху на місяць, декаду, добу та зміну.

Календарний план-графік дільниці може відноситься до одного з основних планів: подетально-пооперaційного, подетального, графік-перелік деталей.

Подетально-пооперaційні плани-графіки складаються в умовах великосерійного виробництва. У серійному виробництві їх доцільно складати при порівняно невеликій номенклатурі деталей на ділянці.

В умовах дрібносерійного, а у ряді випадків і серійного типу виробництва побудова подетально-пооперaційних календарних планів-графіків буває недоцільною. Тому, в цих умовах розробляють узагальнений план-графік. У ньому одиницею планування є не окрема деталеоперація, а деталь, для якої визначають час запуску партії в обробку та випуску з дільниці, тобто здійснюється розробка подетального плану-графіка.

В умовах одиничного, а іноді і дрібносерійного виробництва при виготовленні дрібних та середніх за розміром деталей з невеликою тривалістю виробничого циклу в ряді випадків розробляється графік-перелік деталей.

До завдань внутрішньоцехового оперативного обліку відносяться: визначення виконання плану випуску деталей і складальних одиниць, а також плану виготовлення деталеопераций і складальних одиниць дільницями; облік наявності та надходження заготовок, матеріалу, незавершеного виробництва на дільниці, цеха; облік браку та використання обладнання на дільницях; визначення виконання змінно-добових завдань дільницями і т.п. Для скорочення обсягу робіт, пов’язаних з регулюванням виробництва, визначається оптимальна частота перерахунків календарних графіків [14].

Поточний контроль та регулювання виробництва відносяться до виробничої диспетчеризації. Диспетчеризація є складовою частиною оперативного управління. Вона включає: безперервний облік фактичного ходу робіт з виконання встановленого плану виробництва і змінно-добових завдань; прийняття оперативних заходів щодо попередження та усунення відхилень від плану і перебоїв в ході виробництва; виявлення та аналіз причин відхилень від встановлених планових завдань і календарних графіків виробництва та прийняття оперативних заходів з ліквідації цих причин; координацію поточної роботи взаємозалежних ланок виробництва; організаційне управління оперативної підготовки всього необхідного для виконання змінно-добових завдань та календарних графіків виробництва.

Диспетчеризація виконує безперервний централізований контроль і оперативне регулювання протікання виробництва з метою забезпечення рівномірного і своєчасного виконання плану випуску продукції.

Для ефективного виконання цих завдань необхідно, щоб диспетчеризація базувалася на обґрунтовано складених виробничих програмах і календарних планах-графіках та виконувалася на основі точної та своєчасної інформації.

В одиничному та дрібносерійному виробництві об’єктами диспетчерського контролю є строки виконання робіт за окремими замовленнями. Диспетчерський контроль здійснюється за розробленими цикловими або мережевими планами-графіками виконання замовлень.

При будь-якому типі виробництва незмінними об’єктами диспетчерського контролю залишаються випуск підприємством товарної продукції та забезпечення виробництва всім необхідним. Але все ж регулювання ходу виробництва не повинно здійснюватися за відхиленнями, як це відбувається зазвичай. Регулювання за відхиленнями не є ефективним, тому йому не властивий профілактичний характер. Більш ефективним є напрям, який зводиться до створення виробничих умов, що забезпечують виконання плану-графіка протягом певного, досить тривалого періоду часу.

Підтримання ритмічного ходу виробництва забезпечується попередженням, а в разі виявлення – швидким усуненням відхилень у ході виробництва, що значною мірою залежить від обраного кроку регулювання. На різних машинобудівних підприємствах встановлюється різний крок регулювання. Крок регулювання включає час очікування, отримання та обробки інформації, час аналізу отриманих результатів і прийняття рішень.

Ефективність виконання завдань з організації та управління виробництвом багато в чому визначається забезпеченням його технічними засобами, які повинні забезпечувати збір та обробку інформації з оперативного обліку, контролю та регулювання ходу виробництва і передачу її за допомогою засобів зв’язку у відповідні підрозділи підприємства [14].

1.4 Системи управління виробництвом

Вже багато десятиліть ряд компаній шукають способи спростити управління виробничими процесами. Для цього використовують різні системи управління виробництвом.

Першим кроком в розробці таких систем стало створення концепції MRP (Materials Resource Planning), яка підтримувала планування матеріальних ресурсів для виробництва. Основна мета концепції MRP полягала в мінімізації витрат, пов’язаних зі складськими запасами (в тому числі і на різних дільницях виробництва). В основі цієї концепції лежить поняття BOM (Bill Of Material – специфікація виробів), яка відображає залежність попиту на сировину, напівфабрикати та інші продукти від плану випуску готової продукції. При цьому дуже важливу роль відіграє час, для обліку якого необхідно мати чітке уявлення про технологічний ланцюжок випуску продукції, тобто знати, яка послідовність і тривалість операцій. На підставі плану випуску продукції, BOM і технологічного ланцюжка здійснюється розрахунок потреби в матеріалах до конкретних строків.

Однак у концепції MRP є серйозний недолік, а саме те, що при розрахунку в рамках цієї концепції не враховуються потреби в матеріалах, ні наявні

виробничі потужності, ні їх завантаження, ні вартість робочої сили. Цей недолік був виправлений в концепції MRP II (Manufacturing Resource Planning – планування виробничих ресурсів). Поняття «MRP II» використовується для позначення відмінності «плану виробничих ресурсів» (MRP II) від «плану потреби в ресурсах» (MRP). MRP II дозволяла враховувати і планувати всі виробничі ресурси підприємства – сировину, матеріали, обладнання, персонал і т.д. MRP II забезпечує координацію маркетингу і виробництва.

Маркетинговий, фінансовий і виробничий відділ погоджують загальний, придатний для роботи план, виражений у виробничому плані.

Виробничий план встановлює загальні рівні виробництва і матеріально-виробничих запасів на відповідний горизонт планування строків. Першочергова мета полягає в тому, щоб визначити норми виробництва, які дозволять виконати поставлені в стратегічному бізнес-плані завдання.

Відділи маркетингу і виробництва повинні щотижня і щодня взаємодіяти з метою коригування плану з урахуванням змін (зміна розміру замовлення, скасування замовлення або затвердження відповідну дату поставки), що відбуваються. Зміни такого роду здійснюються в рамках генерального календарного плану виробництва [21].

Завдання, які підлягають виконанню при календарному плануванні виробництва, полягають у наступному [12, 30-33]: вибір ресурсів для виконання усіх поставлених завдань, агрегування та організація упорядкування різних видів ресурсів, детальне планування використання ресурсів (пристроїв) у часі, вирішення проблем упорядкування виконання завдань на кожній із одиниць обладнання, побудова календарних планів роботи обладнання та виконання завдань, календарне планування роботи виробничого транспорту [11].

Оперативно-календарне планування складається з чотирьох функціональних етапів [34]:

1) об’ємно-календарне планування (Master Production Schedule, MPS);

2) баланс виробничих потужностей (Capacity Planning Problem, CPP);

3) розрахунок виробничого розкладу (Production Scheduling, PS);

4) групування деталей і складальних одиниць та обладнання (Group Technology, GT).

Менеджери відділів маркетингу та виробництва можуть вносити зміни в генеральні календарні плани виробництва з урахуванням змін прогнозованого попиту. Управління підприємства може змінювати виробничий план відповідно до загальних змін попиту або положення з ресурсами. Проте, всі співробітники працюють в рамках системи MRP II. Вона служить механізмом координації роботи маркетингового, фінансового, виробничого та інших підрозділів компанії. MRP II являє собою метод ефективного планування всіх ресурсів виробничого підприємства [21].

По мірі розвитку концепції MRP II до неї поступово додавалися можливості обліку інших витрат підприємства. Так з'явилася концепція ERP (Enterprise Resource Planning – планування ресурсів підприємства). В основі ERP лежить принцип створення єдиного сховища даних, що містить всю ділову інформацію, накопичену організацією в процесі ведення бізнесу, зокрема фінансову інформацію, дані, пов'язані з виробництвом, управлінням персоналом, і будь-які інші дані, які стають одночасно доступними для всіх працівників, що володіють відповідними повноваженнями.

У дев'ятому виданні «Словника APICS» Американської Асоціації Контролю над Виробництвом та матеріально-виробничих запасів (APICS) дається таке визначення ERP: призначена для ведення звітності інформаційна система ідентифікації та планування підприємства – глобальних ресурсів, необхідних для виробництва, транспортування і складання звітів за замовленнями клієнтів.

У роботі [35] ERP-система має наступне визначення – це інтегрована система на базі інформаційних технологій для управління внутрішніми і зовнішніми ресурсами підприємства.

Концепція ERP знайшла широке застосування, оскільки планування ресурсів дозволяло скоротити час випуску продукції, знизити рівень товарно-матеріальних запасів, а також поліпшити зворотний зв'язок зі споживачем при

одночасному скороченні адміністративного апарату. Стандарт ERP дозволив об’єднати всі ресурси підприємства і підвищити ефективність управління ними.

Для повноцінної експлуатації повинні бути передбачені програми для планування, календарного планування, калькуляції собівартості та інше на всіх рівнях організації, у робочих центрах, відділах, підрозділах та для всіх їх разом. Важливо відзначити, що ERP охоплює компанію цілком, а MRP II відноситься до виробництва [21]. Проектування та застосування ERP-систем широко описані у роботах [36, 37].

На теперішній час практично всі сучасні західні системи управління виробництвом базуються на концепції ERP і відповідають її рекомендаціям. Ці рекомендації виробляються американською громадською організацією APICS, що об’єднує виробників, консультантів у галузі управління виробництвом, а також розробників програмного забезпечення (ПЗ).

Одним із стандартів систем управління підприємствами є CSRP (Customer Synchronized Resource Planning), крім усього іншого, він охоплює і взаємодію з клієнтами: оформлення нарядів / замовлень і технічних завдань, підтримка замовника на місцях і т.п. Таким чином, якщо стандарти MRP, MRP II та ERP орієнтовані на внутрішню організацію підприємства, то стандарт CSRP включає в себе повний цикл – від проектування майбутнього виробу, з урахуванням вимог замовника, до гарантійного і сервісного обслуговування після продажу. Суть концепції CSRP головним чином полягає в тому, щоб інтегрувати замовника в систему управління підприємством. Відповідно до даної концепції не відділ збуту, а безпосередньо сам покупець розміщує замовлення на виготовлення продукції, сам відповідає за правильність його виконання і при необхідності відстежує дотримання строків виробництва і поставки. При цьому, саме підприємство може дуже чітко відстежувати тенденції попиту на його продукцію.

На сьогодні існує безліч програмних продуктів, які дозволяють вирішувати ті чи інші завдання календарного планування. Наприклад, функціональні етапи календарного планування MPS та CPP виконуються, як правило, системами

класу ERP (Enterprise Resource Planning), етапи PS та GT реалізуються засобами MES (Manufacturing Execution Systems). Хоча для CPP і PS етапів календарного планування виробництва також успішно використовують APS (Advanced Planning Systems) [34].

На передових як вітчизняних так і зарубіжних підприємствах широко впроваджуються:

– системи управління ресурсами підприємства (ERP) [34], що формують в автоматизованому режимі номенклатурні плани виробництва;

– програми для вдосконаленого планування (Advanced Planning & Scheduling, APS) [34], що представляє собою концепцію оптимізованого (або синхронного) виробничого планування, головною особливістю якої є можливість швидкого складання планів з урахуванням наявних ресурсів і виробничих обмежень (переналагодження обладнання, доступність оснащення, зв'язок між пристроями та ін.) і швидкого перепланування за заздалегідь складеним сценарієм оптимізації, концепція APS поєднала в собі основні елементи концепцій MRP, MRP II, Finite Capacity Scheduling (FCS);

– виконавчі виробничі системи (Manufacturing Execution System, MES) [34], що пов'язують воєдино всі бізнес-процеси підприємства з виробничими процесами, які підвищують ефективність використання верстатної системи;

– системи управління трудовими ресурсами (Human Resource Management, HRM) [38], їх ділять на «розрахункові», «облікові» та системи управління трудовими ресурсами за рівнями автоматизації управління персоналом, які відповідають етапам розвитку прикладних програмних рішень для кадрових служб;

– системи для ефективного управління складом (Warehouse Management System, WMS) [39], що підтримують усі складські бізнес-процеси, якісне планування операцій, розробку раціональної топології складу, управління ресурсами, номенклатурою, впровадження передових методів для обробки і зберігання запасів;

– системи управління ланцюгами поставок (Supply Chain Management, SCM) [40], які призначені для автоматизації та управління всіма етапами постачання ресурсами підприємства і для контролю за рухом товару на підприємстві;

– корпоративна інформаційна система управління відносинами з клієнтами (Customer Relationship Management, CRM) [41], що дає можливість не просто автоматизувати взаємодію з клієнтами і процес продажів, а вибудовувати їх роботу таким чином, щоб отримувати максимальний результат;

– система управління життєвим циклом продукції (Product Lifecycle Management, PLM) [42], являє собою методологію застосування сучасних інформаційних технологій для підвищення конкурентоспроможності промислових підприємств. Основними компонентами PLM-системи є:

1) PDM-система (Product Data Management) – система управління даними про виріб, є основою PLM, призначена для зберігання і управління даними;
2) CAD-система (Computer Aided Design) – проектування виробів;
3) CAE-система (Computer Aided Engineering) – інженерні розрахунки;
4) CAPP-система (Computer Aided Production Planning) – розробка прогресивних технологічних процесів;
5) CAM-система (Computer Aided Manufacturing) – розробка керуючих програм для верстатів;
6) MPM-система (Manufacturing Process Management) – моделювання та аналіз виробництва виробів;

– система скорочення витрат (Enterprise Asset Management, EAM) [43] орієнтована на скорочення витрат, пов'язаних із обслуговуванням устаткування і підвищенням продуктивності.

Інтеграція різних систем привела до розмивання меж між ними. Наслідком цього процесу стало нечітке розуміння, навіть у середовищі IT-фахівців, функціональної спеціалізації різних систем планування та управління виробництвом. Виробники ERP-систем стали заявляти, що їхні продукти тепер

«легко вирішують» всі MES-завдання, а виробники MES-систем, у свою чергу, почали переконувати представників виробництва, що саме їх системи прийшли на зміну «застарілим» системам класу ERP. Продавці APS твердять про те, що, системи класу APS легко можуть замінити як ERP, так і MES разом узяті [34].

Розглянемо існуючі функціональні відмінності між декількома розглянутими системах планування виробництва.

Система ERP виконує об'ємно-календарне планування (MPS), попутно вирішуючи завдання балансу виробничих потужностей (CPP).

Система APS уточнює розраховані в ERP планові завдання, розподіляючи (оптимізуючи) їх по робочих місцях. На цьому етапі можливе внесення корекцій у вихідний MPS. Оскільки послідовність цих завдань, як правило, залежить від залученого в контур планування ланцюжка поставок матеріалів і комплектуючих (SCM), то порушувати цю послідовність всередині цеху вже не можна без ризику зруйнувати основний виробничий план.

Система MES за рахунок можливостей багатокритеріальної оптимізації дозволяє, варіюючи пріоритети різних завдань, вирішити задачу максимізації швидкості їх проходження через робочі місця. При цьому допускається не тільки перевпрорядкування завдань, а й перерозподіл деяких з них на інші взаємозамінні робочі місця. Відзначимо, що на MES-рівні ніяких змін у загальний виробничий план вносити не можна [34].

Інформаційна технологія оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк», розробка якої розглядається у даній роботі, є частково інтегрованою системою, яка загалом відноситься до класу ERP-систем, але має у своєму складі елементи APS та MES-систем. ІТОКПДВ, оскільки і ERP-система виконує об'ємно-календарне планування (MPS), попутно вирішуючи завдання балансу виробничих потужностей (CPP), також так само як APS-система розподіляє (оптимізуючи) розраховані планові завдання за дільницями та пристроями, а деякі функції ІТОКПДВ, якими володіє і MES-система дозволяють, вирішувати задачу максимізації швидкості їх проходження через пристрої і не тільки цю, а й задачі

складання допустимих розкладів виконання завдань із спільним директивним строком для паралельних пристроїв з метою максимізації моменту запуску пристроїв та задачі складання допустимих розкладів виконання завдань паралельними пристроями з метою мінімізації максимуму відхилення моментів завершення завдань пристроями від заданого строку, при цьому допускається не тільки перевпрорядкування завдань, а й перерозподіл деяких з них на інші взаємозамінні пристрої.

Отже, важливою складовою ІТОКПДВ є реалізація моделей достатньо складної структури, зокрема моделей теорії розкладів.

1.5 Класифікація задач теорії розкладів

Задачі теорії розкладів виникають там, де є необхідність вибору тієї чи іншої черговості виконання завдань із використанням будь-яких ресурсів. Питання про те, який вид ресурсів використовувати, коли і в якому порядку виконувати множину завдань, впливає на розмір затрат, пов’язаних із їх виконанням, на час завершення всього комплексу завдань, а також і на інші показники ефективності розкладу. Правильно складений розклад дозволяє виконати один і той же комплекс завдань меншим об’ємом ресурсів, що використовуються та за коротший строк.

У найбільш загальному формулюванні задача складання розкладу полягає в наступному. Використовуючи деяку множину ресурсів або обслуговуючих пристроїв повинна бути виконана деяка фіксована система завдань із заданим набором характеристик. Такі характеристики можуть розглядатися як: перелік операцій, які входять до складу кожного із завдань; обмеження на послідовність виконання операцій кожного завдання й часткові послідовності строків їх завершення, а також строки виконання завдань; ресурси, які необхідні для виконання кожної з операцій усіх завдань; час та вартість виконання робіт кожної із операцій завдань при використанні різних видів ресурсів; директивні строки початку та завершення виконання кожного завдання; перелік та

характеристики ресурсів, необхідних для виконання всіх операцій кожного із завдань [11].

Пристрій для якого складається розклад роботи може бути один, а може бути їх декілька та вони можуть працювати паралельно.

У залежності від особливості задачі паралельні пристрої можуть бути:

– ідентичними (identical parallel machines), тобто з одними й тими ж технічними характеристиками;

– одноманітними/пропорційними паралельними пристроями (uniform/proportional parallel machines), тобто ідентичними тільки за функціональними можливостями, але з різною продуктивністю або швидкодією;

– не пов’язаними паралельними пристроями (unrelated parallel machines), тобто бути відмінними за функціональністю [44].

Коло вирішуваних у рамках теорії розкладів задач можна розділити на наступні групи [12, 30-32, 45, 46]:

– задачі розподілу завдань, вибір ресурсів (пристроїв) для їх виконання;

– задачі упорядкування виконання множини завдань на кожному із пристроїв;

– задачі узгодження часу виконання підмножин завдань, що виконуються;

– задачі вибору складу та розміщення обладнання, визначення кількості, об’ємів та розміщення проміжних буферних накопичувачів [47];

– маршрутизація руху виробничого транспорту;

– побудова розкладів проведення занять, руху автомобільного, залізничного, міського транспорту та руху літаків.

Найбільш дослідженим класом задач теорії розкладів є задачі упорядкування [48-52]. У цих задачах обладнання, на якому виконується вся множина завдань, тривалості цих завдань та час переналаштувань пристроїв

передбачуються заданими. Необхідно знайти найефективніші послідовності виконання завдань на кожному із пристроїв.

Найбільше розповсюдження серед цього класу проблем отримали задачі визначення черговості виконання завдань на одному пристрої – Single-machine-Problem або one-machine sequencing problem [32, 53-56]. У найпростішій постановці задачі передбачається тільки один пристрій або група пристроїв, на яких необхідно виконати множину завдань, кожне з яких включає тільки одну операцію. У випадку наявності декількох паралельних пристроїв з однаковими або різними технічними характеристиками [32, 57-59] та за умови, що кожне із завдань може виконуватися на будь-якому із пристроїв, то задача, що розглядається відноситься як до класу розподільчих, так і до класу задач упорядкування.

До цього класу задач упорядкування в теорії розкладів також відносяться задачі визначення черговості виконання множини завдань для групи розміщених один за одним пристроїв (конвеєр) [32, 48, 60] (Flop-Shop-Problem), Flop-Shop-Problem найбільш характерна для машинобудівного виробництва, харчової, текстильної та інших галузей промисловості.

При наявності декількох пристроїв та якщо кожне завдання, яке повинно на них виконуватися, складається із множини операцій та послідовності виконання множини операцій кожного із завдань відмінні один від одного, то виконання кожної із цих операцій можливе тільки на одному строго фіксованому пристрої (Job-Shop-Problem), задача, що розглядається, може бути віднесена як до класу розподільчих, так і до класу упорядкування задач теорії розкладів [12, 30, 32, 56, 61-65].

У задачах узгодження необхідно узгодити в часі множину завдань, що виконуються кожною групою використовуваних ресурсів. Типовим представником такого класу задач є проблеми синхронізації роботи конвеєрної лінії [32, 47].

Також задачі теорії розкладів можна розділити на дві групи: задачі без переривань, в яких час виконання кожної операції всіх розглянутих завдань не допускає розривів під час їх виконання; задачі з перериваннями [66].

При формуванні критеріїв оптимальності розкладів повинні бути враховані наступні фактори: витрати, які пов’язані з порушенням директивних строків завершення виконання завдань; витрати, які пов’язані з неефективною роботою обладнання: непродуктивні простої пристроїв, втрати на переналагодження при переході від виконання попередньої операції до наступної.

Критеріями оптимальності розкладів можуть бути використані різні приведені нижче економічні показники [12, 30-32]:

- критерії оптимальності, орієнтовані на ринок та пов’язані з задоволенням потреб замовників:
  1) мінімальний середній та середньозважений час очікування виконання всіх завдань;
  2) мінімальний середній та середньозважений час виконання всіх завдань;
  3) мінімальний середній або середньозважений час завершення виконання всіх завдань;
  4) мінімальний час виконання розкладу;
  5) мінімальне фактичне та середньозважене число не виконаних у директивний строк завдань;
  6) мінімальний час відхилення від директивного строку завершення виконання серед всієї множини завдань.
- критерії оптимальності, орієнтовані на ефективність використання ресурсів:
  1) сумарний, середній та середньозважений час очікування пристроями початку виконання завдань;
  2) сумарний, середній та середньозважений час роботи пристроїв, необхідний для виконання всіх завдань розкладу [11].

Задачі складання розкладу за типом режиму планування поділяються на: в автономному (офлайн) режимі планування та в режимі онлайн планування.

В автономному режимі планування час звільнення пристроїв, час обробки завдань, директивний строк та інші необхідні дані для виконання кожного із завдань відомі до того, як визначено план виконання завдань на одноманітних/пропорційних паралельних пристроях. Онлайн режим планування здійснює планування роботи пристроїв на основі характеристик, що надходять.

Завдання, які надходять на виконання, можуть бути класифіковані як без переривань та з перериваннями. Якщо завдання, закріплене за пристроєм, продовжує оброблюватися на ньому до свого повного завершення, то воно називається завданням без переривань. Якщо ж завдання, яке призначене на пристрій припиняється або переназначається на інший пристрій, то воно називається попереджувальним завданням [44].

У даній роботі будуть вирішуватися задачі складання розкладів, які матимуть наступні характеристики: у систему одночасно надходить певна кількість завдань, які мають визначену тривалість виконання; завдання виконуються паралельними пристроями, які мають ідентичні функціональні можливості, мають однакову або різну продуктивність, тобто різну швидкість виконання завдань (тобто пристрої можна впорядкувати за швидкістю виконання завдання й цей порядок однаковий для всіх завдань, при цьому для кожного пристрою задається коефіцієнт такий, що тривалість виконання завдання на цьому пристрої пропорційний цьому коефіцієнту, у літературі такі пристрої іноді називають одноманітними або пропорційними); завдання виконуються пристроями без переривань; задачі складання розкладу вирішуються в автономному режимі; для паралельних пропорційних пристроїв необхідно скласти розклад виконання завдань, що дозволяє досягнути ефективного використання ресурсів відповідно заданого критерію.

1.6 Методи вирішення задач теорії розкладів

Математичні методи вирішення задач теорії розкладів можна поділити на наступні види:

– математичні методи цілочисельного математичного програмування [12, 32, 57, 67, 68];

– математичні методи задач теорії розкладів у вигляді побудови допустимих екстремальних шляхів на графі [12, 32, 56, 69, 70].

Для вирішення задач теорії розкладів розроблено велику кількість методів дискретної оптимізації, які умовно можна розділити на дві групи: точні та наближені. Умовність цього поділу витікає із того, що багато точних методів можуть застосовуватися як наближені, а наближені методи при певних умовах можуть застосовуватися як точні або їх складова частина [71].

Найбільшого розповсюдження отримали наступні підходи та методи [11]:

– точні методи:
  1) методи цілочисельного програмування:
     a) лінійне цілочисельне програмування;
     b) нелінійне цілочисельне програмування; булеве програмування;
  2) послідовні алгоритми оптимізації:
     a) метод гілок та меж;
     b) динамічне програмування та методи аналізу і відсіву варіантів;
     c) методи теорії графів;
– наближені методи:
  1) обмеження об’єму розрахунків в послідовних алгоритмах оптимізації;
  2) методи випадкового пошуку:
     a) методи глобального випадкового пошуку,
     b) методи локальної варіації;
  3) генетичні алгоритми та еволюційні стратегії;
  4) евристичні методи;

5) гибридні алгоритми;
6) метаевристичні методи;
7) імітаційне моделювання.

Широкого застосування набула ПДС-методологія для розв'язання важкорозв'язуваних задач комбінаторної оптимізації.

Реальні практичні задачі теорії розкладів відносяться до класу NP-повних задач. Практично, це складно розв'язувані задачі й для алгоритму, який вирішить NP-повну задачу, буде потрібно, у найгіршому випадку, експоненційна кількість часу для отримання точного рішення. Відповідно, такий алгоритм може використовуватися на практиці, тільки для отримання точних рішень задач невеликої розмірності [11]. Через великий обсяг практичних задач ні один із розглянутих вище підходів не гарантує отримання точного їх вирішення у прийнятний час. Ця обставина зумовила розробку наближених методів, які дозволяють отримати прийнятне рішення при порівняно не великих затратах часу.

До наближених методів рішення відносяться методи глобального випадкового пошуку і локальних варіацій [72-75]. Характерною особливістю цього методу є те, що на кожній ітерації процес вибору нового рішення виконується не детерміновано, а в результаті реалізації деякого випадкового процесу, при чому стратегія вибору може залежати від передісторії пошуку. Одним із розвитком методів локальної варіації є Tabu-Search-аналіз [76].

До наближених методів також відносяться генетичні алгоритми та еволюційні стратегії. Генетичні алгоритми детально описані у публікаціях [77-85] та є евристичними алгоритмами пошуку, які використовуються для рішення задач оптимізації та моделювання послідовності підбору, комбінації та варіації шуканих параметрів на основі механізмів, які нагадують біологічну еволюцію. Оскільки, алгоритм в процесі пошуку використовує деяке кодування множини параметрів замість самих параметрів, то він може ефективно застосовуватися для рішення задач теорії розкладів, дискретної оптимізації, визначених як на числових множинах, так і на кінцевих множинах вільної природи.

Достатньо широке розповсюдження в наш час отримали різні евристичні методи рішення задачі [32, 45, 86-92]. Основні підходи при побудові цих методів базуються на прийомах зниження вимог та використанні різних видів вирішальних правил, обґрунтованих при рішенні близьких за постановкою задачі, але які не містять цілої множини додаткових складностей та обмежень, характеристик для реально розглянутої проблеми. Евристичні алгоритми використовують різні підходи без суворих обґрунтувань [11, 93, 94]. Евристичні методи дозволяють знаходити прийнятні рішення навіть у дуже складних випадках: при неповноті, випадковості вихідних даних, відсутності адекватної математичної моделі, NP-складності вирішуваної задачі та відсутності точних методів їх рішення.

Також часто для отримання наближеного рішення задачі використовується «лівостороння» схема розгалуження у методі гілок та меж [95, 96] із зупинкою алгоритму у випадку отримання першого допустимого рішення задачі.

Більшість задач теорії розкладів можуть бути сформульовані як задачі цілочисельного математичного програмування. Частковим випадком яких є задача лінійного цілочисельного програмування [67]. Динамічне програмування [97-101] зазвичай застосовується до вирішенню задач, які можна розбити на послідовність задач меншої розмірності та більш простої структури.

Потреба у розробці алгоритмів складання розкладів, що забезпечать високу якість отримуваних результатів і не будуть потребувати значних обчислювальних ресурсів потребує застосування методології побудови ПДС-алгоритмів для важкорозв’язуваних задач комбінаторної оптимізації [102] (ПДС означає поліноміальна декомпозиційна складова). Зміст цієї методології полягає у наступному. Спочатку на основі теоретичного аналізу досліджуваної задачі виявляються умови оптимальності допустимих розв’язків, потім розробляється алгоритм розв’язання задачі, що має дві складові: поліноміальну і експоненційну. Поліноміальна складова породжується логіко-аналітичними умовами (p – умовами), виконання яких гарантує оптимальність отриманого розв’язку і синтезується таким чином, щоб послідовна процедура конструювання

допустимих розв'язків була найбільш ефективна з точки зору реалізації p – умов. Коли допустимий розв'язок, отриманий поліноміальною складовою, не задовольняє p – умовам, то розв'язання задачі може продовжуватись експоненційною складовою алгоритму.

Оскільки, у роботі будуть проводитися дослідження задач складання розкладів саме для паралельних пристроїв (як ідентичних, так і різної продуктивності), то проаналізуємо сучасні досягнення наукової спільноти щодо вирішення задач цього класу із застосуванням різних математичних методів.

Досягнення різних науковців при вирішенні задачі складання розкладів для паралельних пристроїв детально наведені в роботі [58], порівняльний аналіз деяких таких методів для вирішення мінімаксної задачі складання розкладів наведено в роботі [103] та інших. Для розв'язку задач складання розкладів саме для паралельних пристроїв застосовують такі методи, як випадковий, локальний та вичерпуючий пошук [104-106], метод гілок та меж [107], мурашині алгоритми [108-110], пошук з заборонами [111], ймовірнісні алгоритми [112], генетичні алгоритми [113-120], меметичний алгоритм [121], гібридний алгоритм оптимізації роєм часток [122], алгоритм зозулі [123], 2-наближення [124], бджолиний алгоритм [125], поліноміальний алгоритм [126-128], імітація відпалу [129-132], різні евристичні алгоритми [56, 133-137] та інші.

Також для розв'язку даної задачі пропонується використання гібридного алгоритму, який поєднує в собі генетичний алгоритм та мурашиний алгоритм [133], змішаного алгоритму нелінійного цілочисельного програмування [138], модифікованого алгоритму поєднання ваги та локального пошуку [139], гібридного алгоритму на основі алгоритму методу рою часток та мурашиного алгоритму[140], застосування нейронної мережі [141].

1.7 Опис задач, що розв'язуються при розробці ІТОКПДВ

У роботі розробляється інформаційна технологія оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк». Дана ІТ спрямована на планування рівномірної, ритмічної

взаємоузгодженої роботи всіх виробничих підрозділів підприємства для забезпечення своєчасного виконання планового завдання при економному використанні ресурсів.

Вважається, що на підприємство, яке виробляє деякі види продукції, надходять замовлення (де міститься інформація про види продукції, кількість та директивні строки). Для кожного виду продукції відома технологічна карта. Під технологічною картою [142] розуміється основний документ технологічної документації, в якому плануються технологія виробництва, обсяги робіт, засоби виробництва і робоча сила, необхідна для їх виконання, а також розмір матеріальних витрат.

Відповідно технологічної карти для виготовлення кожного виробу необхідно виконати визначену множину завдань (технологічних операцій), для кожного із завдань відомий час виконання. Якщо надійшло замовлення на новий вид продукції, то фахівці технологічного відділу розробляють для нього технологічну карту та вносять її до системи. Планово-економічний відділ на основі пакету замовлень формує об'ємний план виробництва на вказаний період, на основі якого формується оперативно-календарний план виробництва.

Зазвичай, виробниче підприємство складається з множини виробничих дільниць. Усі дільниці є унікальними в сенсі того, що кожна з них може виконувати лише певний вид завдань, проте різні пристрої дільниці можуть працювати з різною продуктивністю, від якої залежить час виконання завдань даним пристроєм. На кожній дільниці один із пристроїв береться за еталонний, саме для нього у технологічній карті вказується час виконання операцій.

Дільниця має в своєму складі деякий набір обладнання (m пристроїв), на який надходить відповідна множина завдань $J = \{1,2,..,j,.,n\}$. Пристрої працюють паралельно, є взаємозамінними і відрізняються один від одного продуктивністю виконання завдань. При цьому пристрої можна впорядкувати за швидкістю виконання завдання і цей порядок однаковий для всіх завдань: для пристрою i заданий коефіцієнт k_i такий, що тривалість виконання завдання j на пристрої i

дорівнює $k_i p_j$ $(i=\overline{1,m})$. Еталонним будемо називати пристрій з коефіцієнтом продуктивності $k=1$. У цьому сенсі величина p_j є тривалістю виконання завдання j на еталонному пристрої. Величину k_i будемо називати коефіцієнтом продуктивності. Передбачається, що всі завдання множини *J* надходять одночасно та можуть мати спільний жорсткий директивний строк *d*, процес обслуговування кожного завдання протікає без переривань до завершення обслуговування. Усі пристрої працюють без переривань. У залежності від вимог до оперативно-календарного планування для дільниць можуть бути сформульовані різні критерії оптимальності розкладів.

Необхідно, базуючись на об'ємному плані виробництва, з урахуванням директивних строків замовлень, скласти розклад виготовлення виробів і виконати планування роботи виробничих підрозділів підприємства.

Ефективно спланувати роботу всіх виробничих підрозділів підприємства відразу достатньо складно, тому постає питання у декомпозиції такої задачі на менші за розміром підзадачі. При цьому, декомпозиція дозволяє планувати роботу кожного підрозділу (дільниці) окремо, але з урахуванням результатів планування на попередніх етапах. Отже, в результаті виникає низка задач календарного планування для кожної з дільниць, схема взаємозв'яку яких, представлена на рис.1.3. Як було зазначено раніше, дільниці відрізняються одна від одної типами, кількістю та характеристиками обладнання. З математичної точки зору методи розв'язання задач повинні враховувати ці відмінності.

Таким чином, у дисертаційній роботі поставленні та розв'язуються наступні задачі теорії розкладів:

1. Складання допустимих розкладів виконання завдань із спільним директивним строком для паралельних пристроїв з метою максимізації моменту запуску пристроїв:
   1.1. Задача складання допустимого розкладу виконання завдань паралельними ідентичними пристроями з метою визначення максимально пізнього моменту початку виконання завдань із спільним директивним строком;

1.2. Задача складання допустимого розкладу виконання завдань паралельними пристроями різної продуктивності з метою визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним строком;

2. Складання допустимих розкладів виконання завдань паралельними пристроями з метою мінімізації максимуму відхилення моментів завершення завдань пристроями від заданого строку:

2.1. Задача складання розкладу виконання завдань паралельними ідентичними пристроями з метою мінімізації максимуму відхилення від директивного строку моментів завершення завдань пристроями;

Задача складання розкладу виконання завдань паралельними пристроями різної продуктивності з метою мінімізації максимуму відхилення від директивного строку моментів завершення завдань пристроями.

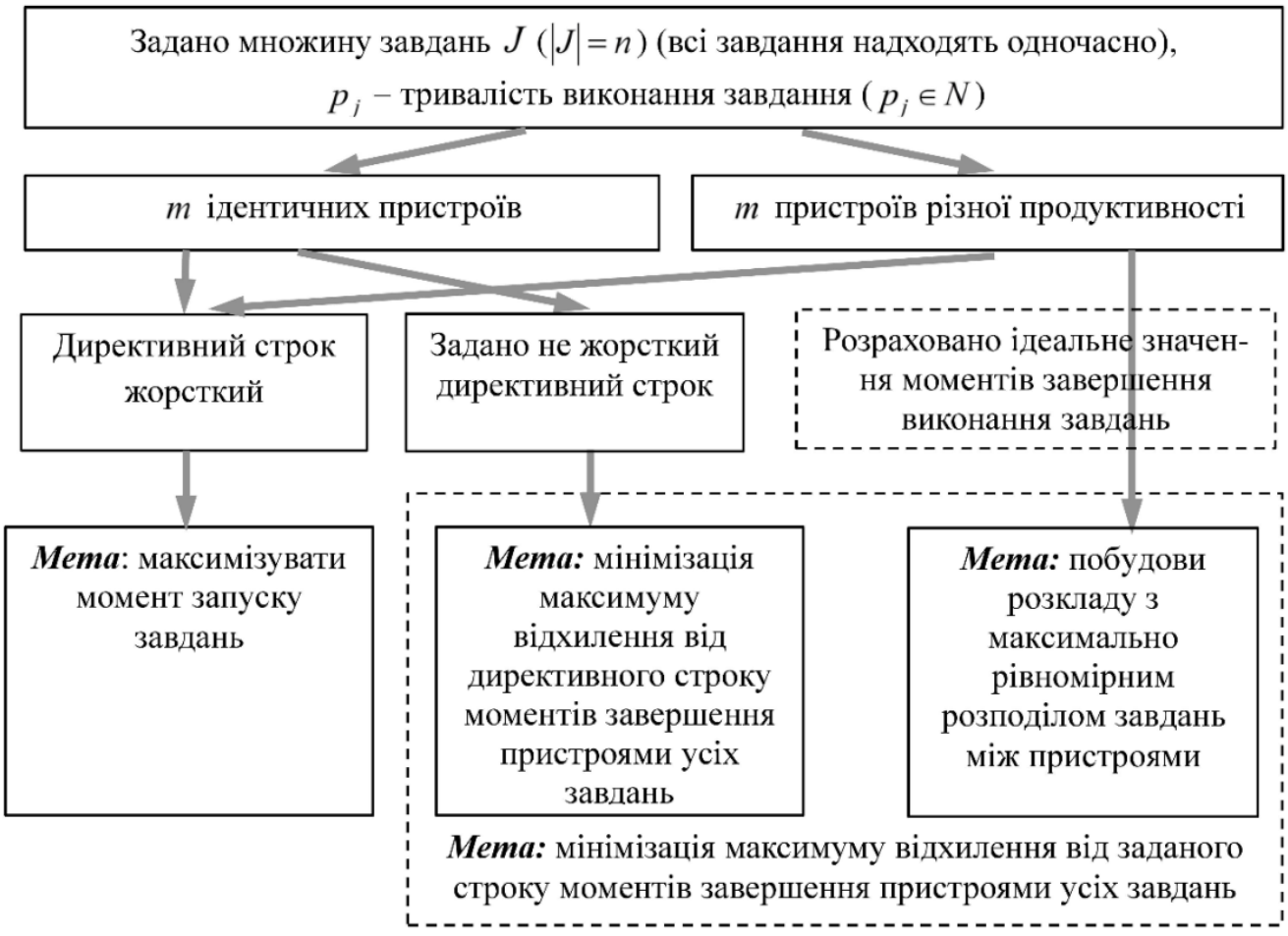


Рис. 1.3. Схема взаємозв’язків задач складання розкладів

Для складання допустимих розкладів виконання завдань із спільним директивним строком для паралельних пристроїв з метою максимізації моменту запуску пристроїв вирішуються дві задачі: для ідентичних пристроїв та пристроїв різної продуктивності.

Для складання допустимих розкладів виконання завдань паралельними пристроями з метою мінімізації максимуму відхилення моментів завершення завдань пристроями від заданого строку також вирішуються дві задачі: для пристроїв однакової та різної продуктивності.

Ці задачі близькі за змістом: у першій задачі величина, відносно якої вимірюється відхилення, є заданою (загальний директивний строк завдань), а в другій ця величина є розрахунковою (представляє собою ідеальне значення моменту завершення виконання усіх завдань). Друга задача є більш узагальненою. Якщо в ній пристрої будуть ідентичними і метою буде мінімізація максимуму відхилення моментів завершення завдань пристроями від заданого загального директивного строку, то отримаємо першу задачу.

Розв’язання наведених задач необхідне для виконання ефективного планування роботи підприємства, тому їх розробка виконується в рамках створення ІТОКПДВ. Під час створення якої був реалізований інструментальний комплекс, який володіє наступними функціями:

а) ведення інформації для виконання планування виробництва;

б) ведення об’ємного плану виробництва;

в) складання плану завантаження потужностей:

1) формування узагальненого технологічного графу проходження завдань по виробничих дільницях;

2) диспетчеризація складання розкладів виконання завдань на окремих дільницях за узагальненим технологічним графом;

3) складання розкладу виконання завдань для виробничої дільниці;

4) складання розкладу виконання завдань для окремого пристрою виробничої дільниці;

г) формування звітності:

1) формування календарного плану виробництва у розрізі виробів (для окремого виробу);
2) формування календарного плану роботи обладнання (для окремої дільниці);
3) графіки відображення фактичного виконання плану, тощо.

Висновок до розділу 1

У розділі описано огляд та застосування систем планування і контролю виробництва для організації рівномірної, ритмічної взаємоузгодженої роботи всіх виробничих підрозділів підприємства. Наведено класифікацію різних типів організації виробництва та опис характеристики дрібносерійного виробництва. Розглянуто методології та парадигми з управління виробництвом, наведена порівняльна характеристика наступних методологій: Ощадне виробництво, Швидкореагуюче виробництво та Активне виробництво, яка показала, що розглянуті методології відрізняються в першу чергу стратегічною орієнтацією. З урахуванням особливостей дрібносерійного типу виробництва та поставленої мети розробка ІТОКПДВ буде виконуватися в межах виробничої системи «точно в строк», яка відноситься до методології LM та може за необхідності забезпечити гнучку перебудову виробництва.

Розглядається оперативне управління основним виробництвом, основною складовою якого є оперативно-календарне планування та диспетчеризація, для яких і розробляється інформаційна технологія.

Представлено огляд програмних систем для планування роботи та управління виробництвом. Визначено, що ІТОКПДВ, розробка якої розглядається у даній роботі, є частково інтегрованою системою, яка загалом відноситься до класу ERP-систем, але має у своєму складі елементи APS та MES-систем.

Було наведено, що важливою складовою ІТОКПДВ є реалізація моделей теорії розкладів. Наведено класифікацію задач теорії розкладів та описані їх

складові та умови вирішення. Розглянуто методи вирішення задач теорії розкладів та проведений детальний аналіз застосування цих методів для реалізації схожих задач. У роботі наведено огляд результатів застосування розповсюджених методів розв’язання задач теорії розкладів. При проведенні аналізу літературних джерел було виявлено роботи, що пропонують різні підходи до розв’язання досліджуваних задач теорії розкладів.

Наведено вимоги до інформаційної технології, яка розробляється в даній роботі.

РОЗДІЛ 2 СКЛАДАННЯ ДОПУСТИМИХ РОЗКЛАДІВ ВИКОНАННЯ ЗАВДАНЬ ІЗ СПІЛЬНИМ ДИРЕКТИВНИМ СТРОКОМ ДЛЯ ПАРАЛЕЛЬНИХ ПРИСТРОЇВ З МЕТОЮ МАКСИМІЗАЦІЇ МОМЕНТУ ЗАПУСКУ ПРИСТРОЇВ

Розглядається задача визначення максимально пізнього моменту початку виконання в допустимому розкладі завдань із спільним директивним строком паралельними пристроями. На практиці пристрої можуть бути як ідентичними так і мати різні продуктивності (рис. 2.1).

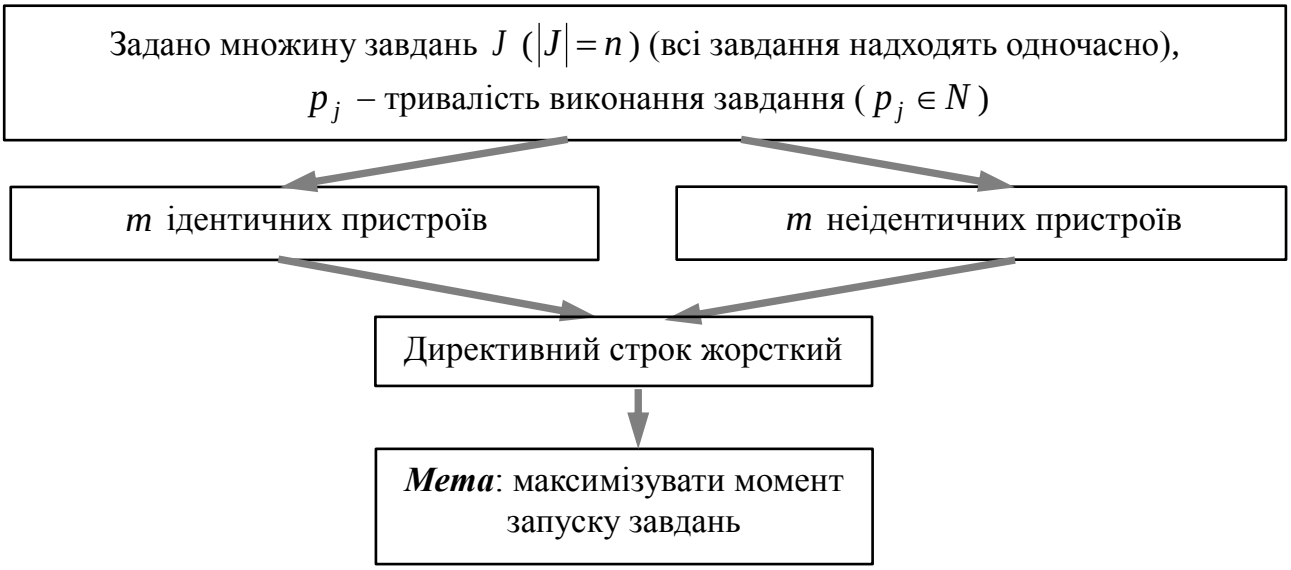


Рис. 2.1. Постановка задач розділу 2

У розділі спочатку досліджується задача складання розкладу роботи паралельних ідентичних пристроїв, а потім – задача з паралельними одноманітними (пропорційними) пристроями. Перша задача є частковим випадком другої. Для цих задач не відомий поліноміальний алгоритм розв’язання.

Ці задачі вже досліджувалися, про що свідчать роботи [143-159 та інші]. Для їх вирішення були запропоновані різні методи, а саме: жадібний локальний пошук [146], табу пошук [148], метод гілок та меж [143, 149], гібридний генетичний алгоритм з локальним пошуком [145], мурашині алгоритми [110], пошук з заборонами [111], генетичні алгоритми [116, 117, 120], меметичний алгоритм [121], гібридний алгоритм оптимізації роєм часток [122],

поліноміальний алгоритм [127] та інші.

Задачі, що розглядаються відрізняються від наведених робіт тим, що запропоновано підхід, який базується на ПДС-методології, згідно якої для цих задач визначені достатні умови оптимальності, які покладені в основу алгоритмів розв'язання. Це дозволяє: по-перше, скоротити кількість ітерацій знаходження розв'язку, по-друге, при виконанні достатніх умов оптимальності можна однозначно стверджувати про оптимальність отриманого розв'язку та по-третє, при не виконанні цих умов визначається величина, на яку, в найгіршому випадку, відрізняється величина отриманого значення критерію від оптимального.

2.1 Задача складання допустимого розкладу виконання завдань паралельними ідентичними пристроями з метою визначення максимально пізнього моменту початку виконання завдань із спільним директивним строком

2.1.1 Постановка задачі

Задано множину завдань J ($|J| = n$), кількість пристроїв m, для завдання $j \in J$ відома тривалість виконання p_j, $p_j \in N$ (N – множина натуральних чисел). Передбачається, що всі завдання надходять одночасно і мають спільний директивний строк d ($d \in N$), процес обслуговування кожного завдання протікає без переривань до завершення обслуговування всіх завдань. Процес виконання завдань кожним з пристроїв є безперервним: після виконання першого по порядку відразу ж починає виконуватися друге і т.д. Усі пристрої починають свою роботу в один і той же момент часу (далі – *момент r запуску* пристроїв).

Необхідно побудувати розклад виконання завдань множини J, у якого всі завдання завершуються до директивного строку і момент r запуску завдань на виконання є максимальним.

Допустимим розкладом будемо називати розклад, в якому всі завдання не запізнюються, і при цьому максимальний час завершення завдань співпадає з директивним строком.

Результати, які були отримані для цієї задачі приведені у роботах [1-3, 13, 161, 162].

2.1.2 Дослідження властивостей задачі

Розглянемо довільний допустимий розклад σ. Позначимо через $T_i(\sigma)$ сумарну тривалість виконання завдань на пристрої i, $i=\overline{1,m}$, вона дорівнює: $T_i(\sigma)=\sum_{j=1}^{n_i} p_{i_j}$, де p_{i_j} – тривалість i_j завдання, яке j-им виконується на i-му пристрої, n_i – кількість завдань, які виконуються на i-му пристрої, $\sum_{i=1}^{m} n_i = n$.

Нехай $T^*(\sigma)=\max_i T_i(\sigma)$. Позначимо через $T^* = \min_{\forall\sigma\in\Omega} T^*(\sigma) = T^*(\sigma^*)$, де Ω – множина всіх допустимих розкладів. У цьому випадку, в допустимому розкладі σ^* момент запуску пристроїв є максимально пізнім і дорівнює $r_{\max}=d-T^*$ та для будь-якого іншого допустимого розкладу значення моменту запуску не може бути більше. Якщо $r_{\max}<0$, то це означає, що допустимого розкладу не існує. Будь-який допустимий розклад σ, у якого $T^*(\sigma)=T^*$, а момент запуску пристроїв $r=d-T^*$, є оптимальним [160].

Нехай $C^*=\left\lfloor \frac{\sum_{j=1}^{n} p_j}{m} \right\rfloor$ (тут $\lfloor a \rfloor$ – найбільше ціле, для якого виконується $\lfloor a \rfloor \le a$), $\delta=\sum_{j=1}^{n} p_j - C^* m$ (за визначенням $\delta \ge 0$).

Розглянемо деякий розклад σ. Введем позначення:

$C_i(\sigma)$ – момент завершення виконання всіх завдань пристроєм i, $i=\overline{1,m}$ ($C_i(\sigma)=r+T_i(\sigma)$, $i=\overline{1,m}$);

$\Delta_i(\sigma)=\max\{0;T_i(\sigma)-C^*\}$, $i=\overline{1,m}$,

$R_i(\sigma)=\max\{0;C^*-T_i(\sigma)\}$, $i=\overline{1,m}$ (далі $R_i(\sigma)$ будемо називати *резервом*

пристрою i, $\Delta_i(\sigma)$ – *виступом* пристрою i);

$I_\Delta(\sigma)$ – множину пристроїв, в яких $\Delta_i(\sigma) > 0$;

$I_R(\sigma)$ – множину пристроїв, в яких $R_i(\sigma) > 0$;

$I_0(\sigma)$ – множину пристроїв, у яких $\Delta_i(\sigma) = R_i(\sigma) = 0$;

$J_i(\sigma)$ – множину завдань, які в розкладі σ виконуються пристроєм i.

Із визначення величин $R_i(\sigma)$ і $\Delta_i(\sigma)$ слідує, що $R_i(\sigma)\Delta_i(\sigma) = 0$, $i = \overline{1,m}$.

З урахуванням позначень цільова функція задачі має вигляд:

$$r(\sigma) = d - \left(C^* + \max_i \Delta_i(\sigma) \right) = d - C^* - \max_i \Delta_i(\sigma).$$

Оскільки, величини d і C^* не залежать від упорядкування, максимізація моменту запуску $r(\sigma)$ еквівалентна мінімізації величини максимального із виступів $\max_i \Delta_i(\sigma)$:

$$r(\sigma) \to \max \quad \sim \quad \max_i \Delta_i(\sigma) \to \min .$$

З урахуванням цього, далі при графічній ілюстрації розкладів будемо вважати, що момент r запуску пристроїв дорівнює нулю.

Задача, що розглядається може бути сформульована, як задача цілочислового лінійного програмування:

$$T_{\max} \to \min ; \ \sum_{i=1}^{m} x_{ij} = 1, \ \forall j \in J ;$$

$$\sum_{i=1}^{n} p_{ij} x_{ij} \le T_{\max}, \ i = \overline{1,m}; \ x_{ij} \in \{0;1\}, \forall j \in N, \ i = \overline{1,m}.$$

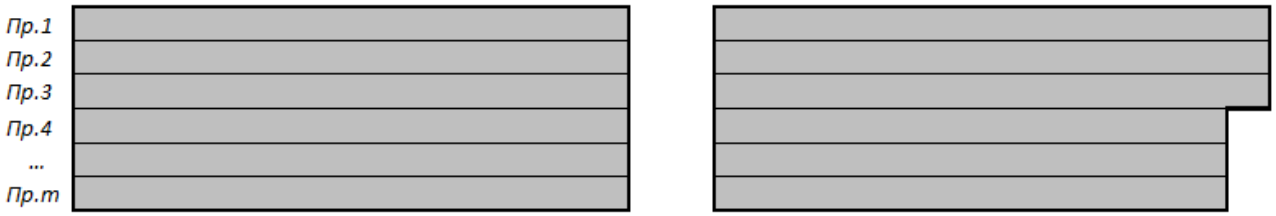
2.1.2.1 Достатні умови оптимальності розкладів

Згідно теорії ПДС-алгоритмів сформулюємо достатні умови оптимальності (ДУО) розкладів.

Достатня умова оптимальності 1. Допустимий розклад, у якого: $\delta = 0$ та $T_i(\sigma) = C^*$, $i = \overline{1,m}$, є рівномірним і, відповідно, оптимальним, оскільки у нього T^* є мінімально можливим [3, 172].

Достатня умова оптимальності 2. Допустимий розклад, у якого $\delta > 0$ є оптимальним, якщо $\forall\ T_i(\sigma)$ (де $T_i(\sigma)$, $i = \overline{1,m}$ – сумарний час роботи *i*-го пристрою) виконується: для будь-яких $i \neq j$, $i, j = \overline{1,n}$, $|T_i(\sigma) - T_j(\sigma)| = 0 \vee b$, де b – довільне раціональне число таке, що $\forall i = \overline{1,m}$ числа p_j / b є цілими, $p_j > 0, j = \overline{1,n}$ [3, 172].

По суті, ДУО дають контур ідеального розкладу – розкладу, який в ідеальному випадку можна отримати для множини завдань з сумарною тривалістю $\sum_{j=1}^{n} p_j$. На рис. 2.2а наведено контур ідеального розкладу для ДУО 1, а на рис. 2.2б наведено контур ідеального розкладу для ДУО 2. Інше тлумачення контуру оптимальності: цей контур відповідає оптимальному розкладу виконання одиничних завдань кількість яких становить $\sum_{j=1}^{n} p_j$.



а) б)

Рис. 2.2. Контури ідеальних розкладів

2.1.2.2 Дослідження властивостей множини допустимих розв'язків

Позначимо через Ψ клас розкладів, для яких виконується:

$$\neg\exists h,\ j, s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma), s \in I_R(\sigma),\ p_j \le R_s(\sigma) \qquad (2.1)$$

та

$$\neg\exists h,\ j, s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma), s \in I_R(\sigma),\ p_j \le \Delta_h(\sigma), R_s(\sigma) > 0. \qquad (2.2)$$

Отже, Ψ не містить таких розкладів, для яких виконується наступне:

– на деякому пристрої $h \in I_\Delta(\sigma)$ є завдання, тривалість якого не перевищує резерв деякого іншого пристрою $s \in I_R(\sigma)$;

– на деякому пристрої $h \in I_\Delta(\sigma)$ є завдання, тривалість якого не перевищує виступу цього пристрою, у той час як на деякому іншому пристрою $s \in I_R(\sigma)$ є ненульовий резерв.

Твердження 2.1. Оптимальний розклад належить класу Ψ.

Доведення.

Якщо для σ виконується:

$$\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ \ j \in J_h(\sigma),\ s \in I_R(\sigma),\ p_j \le R_s(\sigma)$$

та/або

$$\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ \ j \in J_h(\sigma),\ \ s \in I_R(\sigma),\ p_j \le \Delta_h(\sigma), R_s(\sigma) > 0,$$

то переміщення завдання j з пристрою h на пристрій s зберігає значення критерію або призводить до його покращення. Покажемо це. Припустимо, що σ – розклад, який не задовольняє умові (2.1). Це означає, що на деякому пристрої $h \in I_\Delta(\sigma)$ є завдання $j \in J_h(\sigma)$, тривалість якого не перевищує резерв деякого пристрою $s \in I_R(\sigma)$: $p_j \le R_s(\sigma)$. Переміщення завдання j з пристрою h на пристрій s приведе до зменшення величини виступу пристрою h:

$$\Delta_h(\sigma^1) = \begin{cases} \Delta_h(\sigma) - p_j, \text{ если } p_j \le \Delta_h(\sigma), \\ 0, \quad \text{если } p_j > \Delta_h(\sigma), \end{cases}$$

залишаючи нульовим виступ пристрою s. Отже, значення критерію в отриманому розкладі σ^1 не погіршиться: $\max_i \Delta_i(\sigma^1) \le \max_i \Delta_i(\sigma)$, а якщо пристрій h мав у розкладі максимальний із всіх виступів, то значення критерію покращиться.

Припустимо, що σ – розклад, який не задовольняє умові (2.2). Це означає, що на деякому пристрої $h \in I_\Delta(\sigma)$ є завдання, тривалість якого не перевищує величини виступу цього пристрою $p_j \le \Delta_h(\sigma)$, у той час як деякий інший пристрій $s \in I_R(\sigma)$ має резерв $R_s(\sigma) > 0$. Переміщення завдання j з пристрою h на пристрій s приведе до зменшення на p_j величини виступу пристрою h, при цьому, виступ пристрою s може як і раніше залишитися нульовим (якщо

$p_j \le R_s(\sigma)$) або прийняти значення $p_j - R_s(\sigma)$ (якщо $p_j > R_s(\sigma)$). Для розкладу σ^1, отриманого в результаті перестановки, виконується: $\max_i \Delta_i(\sigma^1) \le \max_i \Delta_i(\sigma)$. А якщо пристрій h мав у розкладі максимальний із всіх виступів, то перестановка призведе до покращення критерію.

Таким чином, не існує оптимальний розклад, який задовольняє умовам (2.1) і (2.2). Що і потрібно було довести.

З урахуванням твердження 2.1 пошук оптимального розкладу слід проводити в класі розкладів Ψ.

Твердження 2.2. Для довільного розкладу σ виконується:

$$\sum_{i=1}^{m} \Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) + \delta .$$

Доведення. Тривалість зайнятості пристроїв i визначається наступним чином:

$$T_i(\sigma) = C^* - R_i(\sigma) + \Delta_i(\sigma),\ i = \overline{1,m} . \qquad (2.3)$$

З іншого боку:

$$T_i(\sigma) = \sum_{j \in J_i(\sigma)} p_j , \qquad (2.4)$$

де $J_i(\sigma)$ – множина індексів завдань, які виконуються пристроєм i.

З урахуванням (2.4) перепишемо (2.3):

$$\sum_{j \in J_i(\sigma)} p_j = C^* - R_i(\sigma) + \Delta_i(\sigma). \qquad (2.5)$$

Просумуємо (2.5) по i та виконаємо деяке перетворення:

$$\sum_{i=1}^{m} \sum_{j \in J_i(\sigma)} p_j = \sum_{i=1}^{m} (C^* - R_i(\sigma) + \Delta_i(\sigma)),$$

$$\sum_{i=1}^{n} p_j = \sum_{i=1}^{m} C^* - \sum_{i=1}^{m} R_i(\sigma) + \sum_{i=1}^{m} \Delta_i(\sigma),$$

$$\sum_{i=1}^{n} p_j = mC^* - \sum_{i=1}^{m} R_i(\sigma) + \sum_{i=1}^{m} \Delta_i(\sigma),$$

$$\sum_{i=1}^{n} p_j - mC^* = \sum_{i=1}^{m} \Delta_i(\sigma) - \sum_{i-1}^{m} R_i(\sigma),$$

$$\delta = \sum_{i=1}^{m} \Delta_i(\sigma) - \sum_{i=1}^{m} R_i(\sigma),$$

$$\sum_{i=1}^{m} \Delta_i(\sigma) = \delta + \sum_{i=1}^{m} R_i(\sigma).$$

Що і потрібно було довести.

З цього твердження випливає, що нерівність $\sum_{i=1}^{m} \Delta_i(\sigma) < \sum_{i=1}^{m} R_i(\sigma)$ неможлива, враховуючи не від'ємність δ. З урахуванням вище введених позначень перепишемо ДУО до виконання яких потрібно прагнути. Для розкладу $\sigma \in \Psi$ можливі такі взаємовиключні випадки.

Випадок I: $\delta = 0$. У цьому випадку ДУО 1 має вигляд: $T_i(\sigma) = C^*$, $i = \overline{1,m}$.

Випадок II: $\delta > 0$ (побудувати розклад з рівномірним завантаженням пристроїв неможливо). Сформулюємо для цього випадку ДУО. Нехай, b – найбільший спільний дільник (НСД) тривалостей виконання завдань $p_j, j = \overline{1,n}$, тоді розклад, в якому для всіх $i = \overline{1,m}$ виконується: $\Delta_i(\sigma) \in \{0, b\}$ є оптимальним. Якщо поділити величину $p_j, j = \overline{1,n}$ на b, то умова оптимальності в цьому випадку формулюється так: розклад, в якому для всіх $i = \overline{1,m}$ виконується умова $\Delta_i(\sigma) \in \{0,1\}$, є оптимальним (далі будемо вважати, що всі p_j мають НСД, рівний 1).

Із застосуванням методології побудови ПДС-алгоритмів [114] на основі ДУО розкладів визначимо множину операцій обміну над розкладами, які дозволять послідовно покращувати значення критерію.

2.1.3 Розробка множини операцій обміну

Для покращення розкладу необхідно направити зусилля на зменшення максимального з виступів $\max_{i \in I_\Delta} \Delta_i(\sigma)$. Для цього пропонується використовувати

обмін завданнями між двома пристроями: перший пристрій h із множини $I_\Delta(\sigma)$ з максимальним значенням $\max\limits_{i\in I_\Delta}\Delta_i(\sigma)$, а другий – довільний пристрій s із множини $I_R(\sigma)$, $I_\Delta(\sigma)$, $I_0(\sigma)$, ($s \neq h$). При цьому, деяка підмножина завдань з пристрою h (позначимо її $K_h(\sigma)$, $K_h(\sigma)\subseteq J_h(\sigma)$) міняється місцями з деякою підмножиною завдань з пристрою s (позначимо цю підмножину, як $L_s(\sigma)$, $L_s(\sigma)\subseteq J_s(\sigma)$). Далі такий процес будемо називати операцією обміну завданнями між підмножинами $J_h(\sigma)$ та $J_s(\sigma)$. Інколи для визначення такої операції використовується термін «перестановка завдань», але для уникнення його подвійного тлумачення далі будемо використовувати термін «обмін завданнями».

Позначимо через θ різницю між сумами тривалостей завдань, які приймають участь у операції обміну:

$$\theta = \sum_{j\in K_h(\sigma)} p_j - \sum_{j\in L_s(\sigma)} p_j \ .$$

Згідно твердження 2.2, зменшення сумарної величини виступів здійснюється за рахунок зменшення сумарної величини резервів. При цьому, в результаті обміну завдань, застосованих до поточного розкладу σ, в новому розкладі σ^1 для величин $\sum\limits_{i=1}^{m}\Delta_i(\sigma^1)$ та $\sum\limits_{i=1}^{m}R_i(\sigma^1)$ виконується:

$$\sum_{i=1}^{m}\Delta_i(\sigma)-\sum_{i=1}^{m}\Delta_i(\sigma^1)=\sum_{i=1}^{m}R_i(\sigma)-\sum_{i=1}^{m}R_i(\sigma^1).$$

Твердження 2.3. Усі операції обміну завданнями, які зменшують $\max\limits_{i\in I_\Delta}\Delta_i(\sigma)$ збільшують момент r запуску.

Твердження 2.4. Будемо будувати допустимий розклад не перераховуючи момент запуску пристроїв, а перерахуємо його після виконання всіх операцій обміну, які покращують значення критерію.

З огляду на умови виконання та наслідки, операції обміну, що призводять до покращення розкладу (з урахуванням твердження 2.3), можна розділити на чотири типи, які умовно позначимо як ***А***, ***Б***, ***В*** та ***Г***.

Мета операцій обміну типу ***A***: зменшення максимального з виступів $\Delta_h(\sigma) = \max\limits_{i \in I_\Delta(\sigma)} \Delta_i(\sigma)$ за рахунок зменшення резерву одного з пристроїв множини I_R ($R_i(\sigma), i \in I_R$). Умови виконання операції обміну типу ***A***:

$$\theta > 0,$$

$$\theta \le \Delta_h(\sigma), \quad \theta \le R_s(\sigma). \tag{2.6}$$

У результаті обміну отримуємо розклад σ^1, у якого:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta, \quad R_s(\sigma^1) = R_s(\sigma) - \theta.$$

Виступ, що був максимальним у σ, зменшився на θ:

$$\Delta_h(\sigma^1) = \max_{i \in I_\Delta(\sigma)} \Delta_i(\sigma) - \theta.$$

Таким чином, розклад σ^1 не гірший за розклад σ, а за відсутності альтернатив при виборі пристрою h – кращий за σ.

Обміни типу ***A*** розділяються на підтипи в залежності від кількості завдань, що приймають участь в обміні. У подальшому аналізі кількість підтипів обміну обмежимо величиною: $\max\{|K_h(\sigma)|, |L_s(\sigma)|\} = 2$ (як буде показано в п. 2.1.7.2 збільшення кількості підтипів, що використовуються при розрахунках призводить до підвищення ефективності алгоритму, але збільшує час його роботи).

Обмін підтипу ***1-1A***: міняються місцями завдання j_1 з пристрою $h \in I_\Delta(\sigma)$ та завдання j_2 з пристрою $s \in I_R(\sigma)$, завдання j_1 та j_2 задовольняють наступним умовам:

$$\theta = p_{j_1} - p_{j_2} > 0,$$

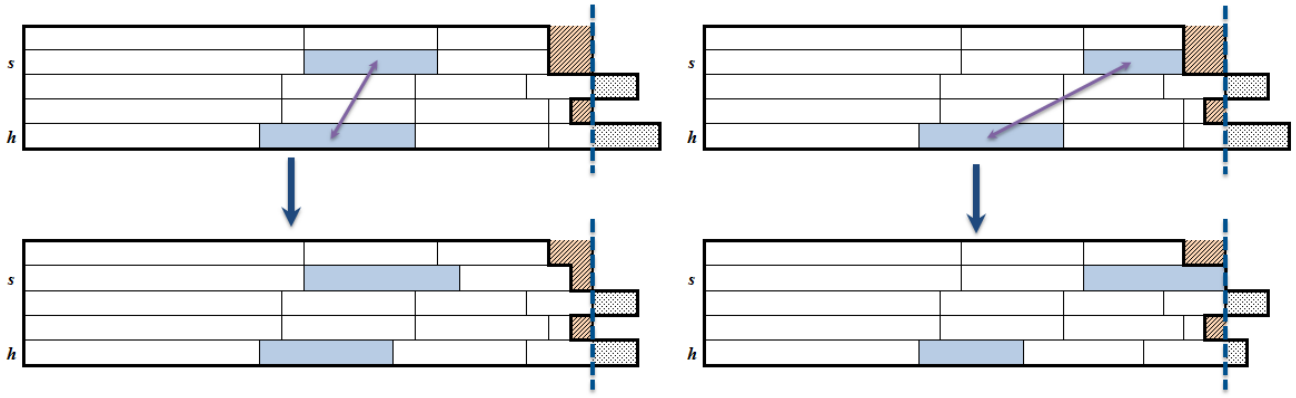
$$p_{j_1} - p_{j_2} \le \Delta_h(\sigma), \tag{2.7}$$

$$p_{j_1} - p_{j_2} \le R_s(\sigma). \tag{2.8}$$

У результаті обміну отримуємо розклад σ^1:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - (p_{j_1} - p_{j_2}), \quad R_s(\sigma^1) = R_s(\sigma) - (p_{j_1} - p_{j_2}).$$

Рисунок 2.3-а) ілюструє обмін підтипу ***1-1A***, для якої (2.6) виконуються як строгі нерівності, а на рисунку 2.3-б) показано перестановку ***1-1A***, у якої передумова (2.7) виконується як строга нерівність, а (2.8) – як строга рівність. На рисунках цього підрозділу та додатку А на рисунках А.1.1-А.1.7 пунктирною лінією позначено значення C^*, відносно якого розраховуються виступи і резерви пристроїв.



а) б)

Рис. 2.3. Обміни підтипу ***1-1A***

Обмін підтипу ***1-2A***: міняються місцями завдання j_1 з пристрою $h \in I_\Delta(\sigma)$ та завдання j_2 та j_3 з пристрою $s \in I_R(\sigma)$, завдання j_1, j_2 та j_3 задовольняють наступним умовам:

$$p_{j_1} > p_{j_2} + p_{j_3}, \quad p_{j_1} - (p_{j_2} + p_{j_3}) \le \Delta_h(\sigma), \quad p_{j_1} - (p_{j_2} + p_{j_3}) \le R_s(\sigma).$$

У результаті обміну отримуємо:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - (p_{j_1} - (p_{j_2} + p_{j_3})),$$

$$R_s(\sigma^1) = R_s(\sigma) - (p_{j_1} - (p_{j_2} + p_{j_3})).$$

Приклад ілюстрації обміну цього підтипу наведено у додатку А1, рис. А.1.1.

Обмін підтипу ***2-1A***: міняються місцями завдання j_1 та j_2 з пристрою $h \in I_\Delta(\sigma)$ та завдання j_3 з пристрою $s \in I_R(\sigma)$, завдання j_1, j_2 та j_3 задовольняють умовам:

$$p_{j_1} + p_{j_2} > p_{j_3},$$

$$(p_{j_1} + p_{j_2}) - p_{j_3} \le \Delta_h(\sigma),$$

$$(p_{j_1} + p_{j_2}) - p_{j_3} \le R_s(\sigma).$$

У результаті обміну отримуємо:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - ((p_{j_1} + p_{j_2}) - p_{j_3}),$$

$$R_s(\sigma^1) = R_s(\sigma) - ((p_{j_1} + p_{j_2}) - p_{j_3}).$$

Приклад ілюстрації обміну цього підтипу наведено у додатку А1, рис. А.1.2.

Обмін підтипу ***2-2А***: міняються місцями завдання j_1, j_2 з пристрою $h \in I_\Delta(\sigma)$ та завдання j_3, j_4 з пристрою $s \in I_R(\sigma)$, завдання j_1, j_2, j_3 та j_4 задовольняють наступним умовам:

$$(p_{j_1} + p_{j_2}) > (p_{j_3} + p_{j_4}),$$

$$(p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4}) \le \Delta_h(\sigma),$$

$$(p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4}) \le R_s(\sigma).$$

У результаті обміну отримуємо:

$$\Delta_h(\sigma) = \Delta_h(\sigma) - ((p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4})),$$

$$R_s(\sigma) = R_s(\sigma) - ((p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4})).$$

Приклад ілюстрації обміну цього підтипу наведено у додатку А1, рисунок А.1.3. Узагальнена характеристика та властивості обмінів типу ***А*** представлені в таблиці 2.2. У результаті обмінів типу ***А*** потужності множин I_Δ та I_R залишаються незмінними або зменшуються. Проте, ця множина обмінів в деяких ситуаціях не дозволяє покращити поточний розв'язок (наприклад, розклад, який наведено на рис. 2.4 не підлягає оптимізації множиною обмінів типу ***А***). Отже, є необхідність розробки операцій обмінів нового типу.

Мета операцій обміну типу ***Б***: зменшення максимального з виступів за рахунок резерву пристрою з множини I_R і з утворенням виступу на цьому пристрої. У результаті обмінів цього типу може збільшитись множина I_Δ.

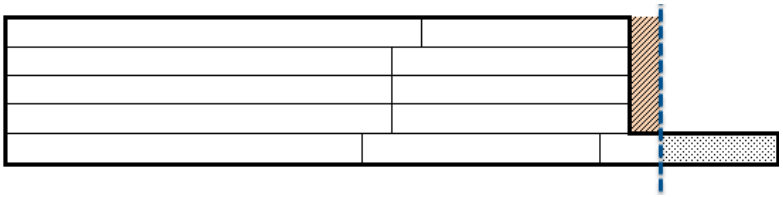


Рис. 2.4. Приклад обмежених можливостей обмінів типу ***А***

Умови виконання обміну типу ***Б***:

$$\theta > 0, \quad \theta \le \Delta_h(\sigma), \quad \theta > R_s(\sigma).$$

У результаті обміну отримуємо розклад σ^1, в якого:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta, \ \Delta_s(\sigma^1) = \theta - R_s(\sigma) > 0,$$

$$s \in I_\Delta(\sigma^1).$$

Виступ, що був максимальним у σ, зменшився на θ:

$$\Delta_h(\sigma^1) = \max_{i \in I_\Delta(\sigma)} \Delta_i(\sigma) - \theta = \Delta_h(\sigma) - \theta,$$

але при цьому для новоутвореного виступу $\Delta_s(\sigma^1)$ виконується: $\Delta_s(\sigma^1) < \Delta_h(\sigma)$. Покажемо це: $\Delta_h(\sigma) - \Delta_s(\sigma^1) = \Delta_h(\sigma) - (\theta - R_s(\sigma)) = \Delta_h(\sigma) + R_s(\sigma) - \theta > 0$, оскільки, $\Delta_h(\sigma) - \theta \ge 0$, $R_s(\sigma) > 0$. Таким чином, розклад σ^1 не гірший за розклад σ, а за відсутності альтернатив при виборі пристрою h – кращий за σ.

Обміни типу ***Б*** також розділяються на підтипи в залежності від кількості завдань, що приймають участь у обміні. Детально розглянемо характеристики обміну підтипу ***1-1Б***.

Обмін підтипу ***1-1Б***: міняються місцями завдання j_1 з пристрою $h \in I_\Delta(\sigma)$ та завдання j_2 з пристрою $s \in I_R(\sigma)$, завдання j_1 та j_2 задовольняють наступним умовам:

$$\theta = p_{j_1} - p_{j_2} > 0,$$

$$p_{j_1} - p_{j_2} \le \Delta_h(\sigma), \quad p_{j_1} - p_{j_2} > R_s(\sigma).$$

Для отриманого в результаті обміну розкладу σ^1 виконується:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - (p_{j_1} - p_{j_2}),$$

$$\Delta_s(\sigma^1) = (p_{j_1} - p_{j_2}) - R_s(\sigma).$$

Приклад ілюстрації обміну наведений у додатку А1, рисунок А.1.4. Узагальнена характеристика та властивості обмінів типу ***Б*** представлені в таблиці 2.2.

Множина обмінів типу ***Б*** призводить до того, що поточна потужність множини I_Δ збільшується, але при цьому величина максимального з виступів зменшується. Наступна множина обмінів за умовами та наслідками є симетричною до множини обмінів типу ***Б***.

Мета операцій обміну типу ***В***: зменшення максимального з виступів за рахунок резерву пристрою з множини I_R з можливим перерозподілом резервів та виступів. В результаті обмінів цього типу змінюється склад множин I_R та I_Δ. На відміну від обмінів типу ***А*** та ***Б***, в обміні цього типу множина $L_s(\sigma)$ може бути порожньою.

Умови виконання обміну типу ***В***:

$$\theta > 0,$$

$$\theta > \Delta_h(\sigma), \quad \theta \geq R_s(\sigma), \quad \Delta_h(\sigma) + R_s(\sigma) > \theta. \tag{2.9}$$

У результаті обміну отримуємо розклад σ^1, у якого:

$$R_h(\sigma^1) = \theta - \Delta_h(\sigma) > 0, \quad \Delta_s(\sigma^1) = \theta - R_s(\sigma).$$

При цьому, в новому розкладі $h \in I_R(\sigma^1)$, $s \notin I_R(\sigma^1)$, $\Delta_h(\sigma) > \Delta_s(\sigma^1)$ (тобто, розклад σ^1 не гірший за σ).

Обмін підтипу ***1-0В***: переставляється завдання j_1 з пристрою $h \in I_\Delta(\sigma)$ на пристрій $s \in I_R(\sigma)$, обмін задовольняє наступним умовам:

$$p_{j_1} > R_s(\sigma), \tag{2.10}$$

$$\theta = p_{j_1} - 0 > 0, \quad p_{j_1} > \Delta_h(\sigma), \quad \Delta_h(\sigma) + R_s(\sigma) > \theta.$$

Нерівність (2.10) є строгою нерівністю на відміну від відповідної умови (2.9), оскільки ми працюємо тільки з розкладами із множини ψ.

Для отриманого в результаті обміну розкладу σ^1 виконується:

$$R_h(\sigma^1) = p_{j_1} - \Delta_h(\sigma) > 0, \quad \Delta_s(\sigma^1) = p_{j_1} - R_s(\sigma) > 0.$$

Приклад обміну наведений у додатку А1, рис. А.1.5.

Обмін підтипу ***1-1В***: міняються місцями завдання j_1 з пристрою $h \in I_{\Delta}(\sigma)$ та завдання j_2 з пристрою $s \in I_R(\sigma)$, завдання j_1 та j_2 задовольняють наступним умовам: $\theta = p_{j_1} - p_{j_2} > 0, \quad p_{j_1} - p_{j_2} > \Delta_h(\sigma), \quad p_{j_1} - p_{j_2} \geq R_s(\sigma).$

Для отриманого в результаті обміну розкладу σ^1 виконується:

$$R_h(\sigma^1) = (p_{j_1} - p_{j_2}) - \Delta_h(\sigma), \quad \Delta_s(\sigma^1) = (p_{j_1} - p_{j_2}) - R_s(\sigma).$$

Приклад обміну наведений у додатку А1, рис. А.1.6.

Характеристики обмінів типу ***В*** представлені в таблиці 2.2.

В таблиці 2.1 розглянуті можливі співвідношення між величинами θ, Δ_h та R_s, а також вказані відповідні їм значення зміни часткової цільової функції (мінімізація максимального із виступів пристроїв h та s).

Таблиця 2.1

Вплив перестановок на значення цільової функції

Умови	**Тип обміну**	**Зменшення значення цільової функції на пристроях *h* та *s***
1	2	3
$\theta < \Delta_h, \quad \theta \leq R_s$	***А***	θ
$\theta = \Delta_h, \quad \theta \leq R_s$	***А***	Δ_h
$\theta \leq \Delta_h, \quad \theta = R_s$ $\theta \leq \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor$	***А***	θ
$\theta \leq \Delta_h, \theta > R_s,$ $\theta \leq \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor$	***Б***	θ
$\theta \leq \Delta_h, \quad \theta > R_s,$ $\theta \geq \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor + 1$	***Б***	$\Delta_h + R_s - \theta$
$\theta > \Delta_h, \ \theta \geq R_s$	***В***	$\Delta_h + R_s - \theta$

Узагальнимо ці результати: для обраної пари пристроїв $h - s$ в результаті операцій обміну типів ***А***, ***Б*** та ***В*** отримуємо таке зменшення значення максимального з виступів цих пристроїв в двох розкладах:

$$\Delta_h(\sigma) - \max\{\Delta_h(\sigma^1), \Delta_s(\sigma^1)\} = \min\{\Delta_h(\sigma),\ \theta, \Delta_h(\sigma) + R_s(\sigma) - \theta\}.$$

Коли вичерпані всі можливості покращення розкладу операціями обміну типами ***А***, ***Б*** та ***В*** пропонується використовувати операції обміну типу ***Г***, які зменшують значення $\max\limits_i \Delta_i(\sigma)$ за рахунок перерозподілу виступів між пристроями. Яскравою ілюстрацією необхідності розробки обмінів цього типу є розклад, представлений на рис. 2.5.

Умови виконання обміну типу ***Г***:

$$\theta > 0, \quad \theta < \Delta_h(\sigma), \quad \theta < \Delta_h(\sigma) - \Delta_s(\sigma).$$

У результаті обміну отримуємо розклад σ^1, в якому:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta, \quad \Delta_s(\sigma^1) = \Delta_s(\sigma) + \theta.$$

Для розкладу σ^1 виконується: $\sum\limits_{i=1}^{m} \Delta_i(\sigma^1) = \sum\limits_{i=1}^{m} \Delta_i(\sigma)$,

але $\max\{\Delta_h(\sigma^1), \Delta_s(\sigma^1)\} < \max\{\Delta_h(\sigma), \Delta_s(\sigma)\}$, тобто, розклад σ^1 не гірший за σ.

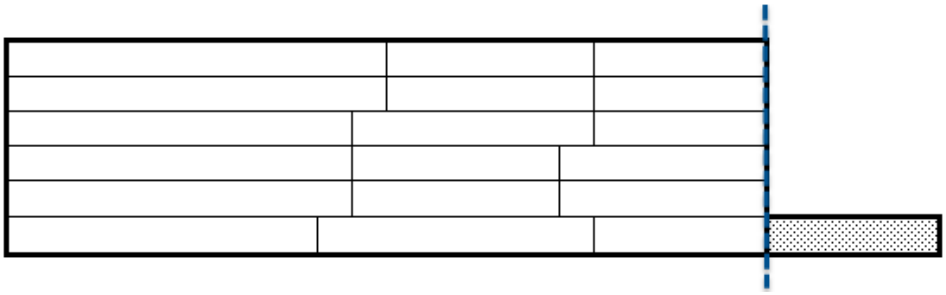


Рис. 2.5. Приклад розкладу, який не можна покращити операціями обміну типами ***А***, ***Б*** та ***В***

Обмін підтипу ***1-1Г***. Міняються місцями завдання j_1 з пристрою $h \in I_\Delta(\sigma)$ та завдання j_2 з пристрою $s \in I_\Delta(\sigma) \cup I_0(\sigma)$ завдання j_1 та j_2 задовольняють наступним умовам:

$$\theta = p_{j_1} - p_{j_2} > 0,$$

$$p_{j_1} - p_{j_2} < \Delta_h(\sigma),$$

$$p_{j_1} - p_{j_2} < \Delta_h(\sigma) - \Delta_s(\sigma).$$

В результаті обміну отримуємо розклад σ^1, в якому:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - (p_{j_1} - p_{j_2}),$$

$$\Delta_s(\sigma^1) = (p_{j_1} - p_{j_2}) + \Delta_s(\sigma).$$

Приклад розкладу з обміном наведено у додатку А1, рис. А.1.7.

Для вибраної пари пристроїв $h - s$ в результаті обмінів типу ***Г*** отримуємо таке зменшення значення максимального із виступів цих пристроїв в двох розкладах:

$$\Delta_h(\sigma) - \max\{\Delta_h(\sigma^1),\ \Delta_s(\sigma^1)\} = \Delta_h(\sigma) - \max\{\Delta_h(\sigma) - \theta,\ \Delta_s(\sigma) + \theta\} =$$

$$= \Delta_h(\sigma) - \min\{\theta;\ \Delta_h(\sigma) - \Delta_s(\sigma) - \theta\}.$$

Характеристики обмінів типу ***Г*** представлені в таблиці 2.2.

Зміна значення цільової функції: $\min|\{\theta;\ \Delta_h(\sigma) - \Delta_h(\sigma) - \theta\}|$.

Ці операцій обміну покладені в основу розробленої поліноміальної складової ПДС-алгоритму рішення задачі.

Таблиця 2.2

Властивості обмінів

Тип операції обміну	**Умови, при яких виконується обмін**	**Характеристики результуючого розкладу σ^1**
1	2	3
А	$\theta > 0,$ $\theta \le \Delta_h(\sigma),$ $\theta \le R_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta,$ $R_s(\sigma^1) = R_s(\sigma) - \theta.$
Б	$\theta > 0,$ $\theta \le \Delta_h(\sigma),$ $\theta > R_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta,$ $\Delta_s(\sigma^1) = \theta - R_s(\sigma)$
В	$\theta > 0,$ $\theta > \Delta_h(\sigma),$ $\theta \ge R_s(\sigma)$	$R_h(\sigma^1) = \theta - \Delta_h(\sigma),$ $\Delta_s(\sigma^1) = \theta - R_s(\sigma)$
Г	$\theta > 0,$ $\theta < \Delta_h(\sigma),$ $\theta < \Delta_h(\sigma) - \Delta_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta,$ $\Delta_s(\sigma^1) = \Delta_s(\sigma) + \theta$

2.1.4 ПДС-алгоритм розв'язання задачі

На основі ДУО та розробленої множини операцій обміну побудований алгоритм визначення максимально пізнього моменту початку виконання завдань паралельними пристроями з спільним директивним строком у допустимому розкладі.

Схема алгоритму

КРОК 1 Побудувати початковий розклад σ^0, $\sigma = \sigma^0$.

КРОК 2 Визначити множини $I_\Delta(\sigma)$ та $I_R(\sigma)$.

КРОК 3 Перевірити виконання достатніх умов оптимальності

ЯКЩО виконується одна із достатніх умов оптимальності

ТО перейти на **КРОК 6** (σ – оптимальний розклад).

КРОК 4 Визначити пристрій h, якому відповідає максимальне значення виступу: $\Delta_h(\sigma) = \max_{i \in I_\Delta} \Delta_i(\sigma)$.

КРОК 5 Виконати для пристрою h, перебираючи всі пристрої $s \in I_R(\sigma) \cup I_0(\sigma) \cup I_\Delta(\sigma)$, покращуючий обмін типу ***А***, ***Б***, ***В*** або ***Г*** (отримати розклад σ^1).

ЯКЩО таких обмінів не знайшлось,

ТО

5.1 Для пристрою h перебираючи всі пристрої $s \in I_0(\sigma) \cup I_\Delta(\sigma)$ виконати обмін типу ***Г*.**

5.2 **ЯКЩО** таких обмінів не знайшлось,

ТО перейти на **КРОК 6**,

ІНАКШЕ перейти на **КРОК 2.**

ІНАКШЕ перейти на **КРОК 2.**

КРОК 6 Визначити максимально пізній момент запуску завдань на виконання в поточному розкладі σ: $r(\sigma) = d - \left(C^* + \max_i \Delta_i(\sigma)\right)$ (згідно твердження 2.3 та 2.4).

КІНЕЦЬ АЛГОРИТМУ

Можливі варіанти реалізації КРОКУ 5:

1) знаходимо перший обмін, який покращує розклад і виконуємо його;

2) перебираємо усі допустимі обміни, знаходимо серед них найкращий та виконуємо його.

Якщо $r_{\max} < 0$, то побудувати допустимий розклад не можливо. У цьому випадку необхідно або зменшити кількість завдань або збільшити кількість пристроїв.

Складність алгоритму складає $O(n^4 W)$, де $W = \sum_{i=1}^{n} p_i$. Це пояснюється наступним. На кожному кроці значення цільової функції (значення максимального із виступів) зменшується, щонайменше на 1. Тому, в найгіршому випадку алгоритм виконує кількість кроків, рівну значенню суми виступів в початковому розкладі σ^0. До початку роботи алгоритму ця величина не відома. Величина W є дуже грубою верхньою оцінкою значення $\sum_{i=1}^{m} \Delta_i(\sigma^0)$. На кожному кроці самою трудомісткою операцією є пошук допустимих обмінів підтипів 2-2, яка зводиться до аналізу тривалостей всіх пар завдань двох обраних пристроїв (верхня межа кількості пар завдань на пристрої дорівнює $n(n-1)/2$, звідси і n^4). Якщо ж використовуються тільки обміни підтипів 1-1, то складність алгоритму наступна: $O(n^2 W)$.

Якщо для результуючого розкладу σ виконується одна з ДУО, то цей розклад є оптимальним. В іншому випадку значення цільової функції розкладу σ відрізняється від її оптимального значення на величину не більше ніж $\max_i \Delta_i(\sigma)$ (якщо $\delta = 0$), $\max_i \Delta_i(\sigma) - 1$ (якщо $\delta > 0$). Тобто ці величини дають верхню межу відхилення значення критерія отриманого розв'язку від оптимального значення.

Алгоритми побудови початкового розкладу

Для побудови початкового розкладу розроблено два алгоритми ***A01*** та ***A02***. Перший з них генерує допустимий розклад випадковим чином, інший – жадібний

алгоритм.

В алгоритмах точка відліку часу співпадає з моментом запуску пристроїв.

Алгоритм *A01* – алгоритм побудови розкладу випадковим чином

КРОК 1 Перенумерувати завдання множини J за порядком зменшення тривалостей їх виконання.

Визначити множину номерів пристроїв $M = \{1, 2, \ldots, m\}$.

КРОК 2 Ініціювати поточну тривалість зайнятості пристроїв: $T_i = 0$, $i = \overline{1,m}$.

КРОК 3 Вибрати завдання $j = 1$.

КРОК 4 Визначити максимальну поточну тривалість зайнятості пристроїв $T_{\max} = \max\limits_{i \in M} T_i$.

КРОК 5 Вибрати пристрій i із множини M з вірогідністю $p_i = \dfrac{T_{\max} - T_i + 1}{\sum\limits_{i \in M}(T_{\max} - T_i + 1)}$ (доданок (+1) введено для того, щоб для пристрою, якому відповідає $C_{\max}$, не була встановлена нульова вірогідність його вибору).

КРОК 6 Призначити завдання j на пристрій i: $T_i = T_i + p_j$.

ЯКЩО $T_i \geq C^*$, **ТО** вилучити пристрій i з розгляду: $M = M \setminus \{i\}$.

КРОК 7 Перейти до наступного завдання $j = j + 1$.

ЯКЩО $j > n$, **ТО** кінець алгоритму,

ІНАКШЕ перейти на **КРОК 4**.

Алгоритм *A02* – жадібний алгоритм побудови розкладу

КРОК 1 Перенумерувати завдання множини J за порядком зменшення тривалостей їх виконання.

КРОК 2 Ініціювати поточну тривалість зайнятості пристроїв: $T_i = 0$, $i = \overline{1,m}$.

КРОК 3 Вибрати завдання $j = 1$.

КРОК 4 Вибрати пристрій i з мінімальною поточною тривалістю зайнятості: $T_i = \max_{1 \le l \le m} T_l$.

КРОК 5 Призначити завдання j на пристрій i: $T_i = T_i + p_j$.

КРОК 6 Перейти до наступного завдання $j = j + 1$.

ЯКЩО $j > n$, **ТО** кінець алгоритму,

ІНАКШЕ перейти на **КРОК 4**.

Очевидно, що розклади, побудовані за допомогою алгоритмів ***A01*** та ***A02***, згідно твердження 2.1, відносяться до класу розкладів Ψ.

Перестановки підтипів 1-1 з точки зору складності розрахунків значно простіші, ніж перестановки підтипів 1-2, 2-1, 2-2. В п. 2.3 наведені результати експериментів, які демонструють як використання обмінів підтипів 1-2, 2-1, 2-2 впливають на роботу алгоритму.

2.2 Задача складання допустимого розкладу виконання завдань паралельними пристроями різної продуктивності з метою визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним строком

2.2.1 Постановка задачі

Задано множину завдань $J = \{1,2,...,n\}$ та кількість пристроїв m. Пристрої працюють паралельно і є взаємозамінними у тому сенсі, що кожний з пристроїв може виконувати будь-яке завдання з множини J. Пристрої відрізняються один від одного продуктивністю виконання завдань і є одноманітними (пропорційними). Тобто, пристрої можна впорядкувати за швидкістю виконання завдання і цей порядок однаковий для всіх завдань: для кожного пристрою i заданий коефіцієнт k_i такий, що тривалість виконання завдання j на пристрої i дорівнює $k_i p_j$. «Еталонним» будемо називати пристрій з коефіцієнтом продуктивності $k = 1$. У цьому сенсі величина p_j є тривалістю виконання завдання j на еталонному пристрої. Величину k_i будемо називати *коефіцієнтом*

продуктивності (якщо $k_i > 1$, то пристрій i менш продуктивний, ніж еталонний, якщо $k_i < 1$ – більш продуктивний).

Передбачається, що всі завдання множини *J* надходять одночасно та мають спільний жорсткий директивний строк *d*, процес обслуговування кожного завдання протікає без переривань до завершення обслуговування завдань. Всі пристрої працюють без переривань. Необхідно знайти максимальний момент запуску пристроїв $r_{\max}$, що дозволяє отримати допустимий розв'язок (розклад, у якому всі завдання не запізнюються). Під моментом запуску розуміється момент початку виконання множини завдань (мінімальний з моментів початку виконання завдань з множини J). Далі, не втрачаючи загальності, будемо вважати, що виконуються такі умови: еталонним є найбільш продуктивний пристрій і у нього $k_i = 1$ (звідси витікає, що усі $k_i \geq 1$); усі p_j є цілими числами.

Задача, що розглядається належить до класу NP. Результати, які були отримані для цієї задачі приведені у роботах [6, 10, 163-166].

2.2.2 Дослідження властивостей задачі

Визначимо теоретично мінімальній час, за який усі пристрої могли б виконати усі завдання в об'ємі $\sum_{j=1}^{n} p_j$. Цю величину можна отримати, якщо вважати, що усі пристрої можуть паралельно виконувати кожне з завдань. Для цього m паралельних пристроїв розглядаються як один, а фактичні тривалості завдань замінюються їх так званими узагальненими значеннями, рівними:

$$\tilde{p}_j = \frac{1}{\sum_{i=1}^{m} \frac{1}{t_{ij}}},$$

де $t_{ij} = k_i p_j$ – тривалість виконання завдання j пристроєм i $(i = \overline{1,m},\ j = \overline{1,n})$;

$\frac{1}{t_{ij}}$ – частка завдання j, яка виконується пристроєм i за одну одиницю часу;

$\sum_{i=1}^{m}\frac{1}{t_{ij}}$ – частка завдання j, яка виконуються всіма пристроями за одну одиницю часу;

$\frac{1}{\sum_{i=1}^{m}\frac{1}{t_{ij}}}$ – тривалість виконання завдання j всіма пристроями.

Тоді, мінімально можливий час, за який усі пристрої могли б виконати усі завдання, становить:

$$C^* = \sum_{j=1`}^{m}\tilde{p}_j\,, \tag{2.11}$$

$$C^* = \sum_{j=1`}^{n}\frac{1}{\sum_{i=1}^{m}\frac{1}{k_i p_j}}. \tag{2.12}$$

Підійдемо до визначення величини C^* з іншого боку. Як і раніше, визначимо C^* – як теоретично мінімально можливий час, за який усі пристрої могли б виконати усі завдання в об'ємі $\sum_{j=1}^{n}p_j$, але тепер за умови, що кожне завдання виконується тільки одним пристроєм. В ідеальному випадку розклад є *рівномірним* – у ньому усі пристрої обслуговують усі свої завдання за час C^*. Очевидно, що рівномірний розклад є оптимальним. Це пояснюється наступним. Переміщення будь-якого завдання з одного пристрою на інший збільшує час завершення усіх завдань на пристрої-реціпієнті. Будь-який обмін груп завдань з двох різних пристроїв також збільшує час завершення усіх завдань на одному з пристроїв.

Введемо величину $c_i^* = \frac{C^*}{k_i}$ – «ідеальну» зведену тривалість зайнятості пристрою i $(i = \overline{1,m})$. У разі рівномірного розкладу справедливо наступне:

$$\sum_{j=1}^{n}p_j = \sum_{i=1}^{m}c_i^*\,, \tag{2.13}$$

(сумарний час виконання усіх завдань в еталонних значеннях тривалостей дорівнює сумі ідеальних зведених тривалостей зайнятості пристроїв)

$$\sum_{j=1}^{n} p_j = \sum_{i=1}^{m} \frac{C^*}{k_i},$$

$$\sum_{j=1}^{n} p_j = C^* \sum_{i=1}^{m} \frac{1}{k_i}.$$

Звідси отримаємо значення C^*:

$$C^* = \frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} \frac{1}{k_i}},$$

$$C^* = \sum_{j=1}^{n} \frac{p_j}{\sum_{i=1}^{m} \frac{1}{k_i}}.$$

Поділимо чисельник та знаменник j-го доданку на величину p_j ($j = \overline{1,n}$):

$$C^* = \sum_{j=1}^{n} \frac{1}{\sum_{i=1}^{m} \frac{1}{k_i p_j}}. \qquad (2.14)$$

Вираз (2.14) співпадає з отриманим раніше виразом (2.12). За умови ідентичності пристроїв ($k_i = 1,\ i = \overline{1,m}$) вираз для величини C^* матиме вигляд $C^* = \frac{\sum_{j=1}^{n} p_j}{m}$ (в цьому випадку результати розрахунків C^* співпадають із результатами, отриманими при дослідженні задачі з ідентичними пристроями).

Величина C^* може бути як цілим так і не цілим числом.

Алгоритм *A03* – алгоритм побудови початкового розкладу

Мета цього евристичного алгоритму – рівномірно навантажити пристрої з урахуванням того, що їх продуктивності різні: поточне завдання призначається на той пристрій, у якого невикористаний зведений фонд робочого часу найбільший.

КРОК 1. Перенумерувати завдання множини J за незростанням тривалостей їх виконання на еталонному пристрої.

КРОК 2. Розрахувати C^* та c_i^* $(i=\overline{1,m})$.

КРОК 3. Встановити значення величин невикористаного зведеного фонду робочого часу пристроїв: $f_i = c_i^*$, $i=\overline{1,m}$.

КРОК 4. Встановити поточну тривалість зайнятості пристроїв: $T_i = 0$, $i=\overline{1,m}$.

КРОК 5. Обрати завдання $j=1$.

КРОК 6. Обрати пристрій i з максимальним невикористаним зведеним фондом робочого часу f_i (а якщо таких декілька, то обрати пристрій з найменшим k_i).

КРОК 7. Призначити завдання j на пристрій i: $T_i := T_i + k_i p_j$; $f_i = f_i - p_j$.

КРОК 8. Перейти до наступного завдання: $j = j+1$. Якщо $j > n$, кінець алгоритму, в іншому випадку переходимо до **КРОКУ 6**.

Зазначимо, що наведений алгоритм за умови ідентичності пристроїв дає той же самий результат, що і алгоритм побудови початкового розкладу ***A02*** для задачі з ідентичними пристроями.

Розглянемо деякий допустимий розклад σ. Позначимо в цьому розкладі:

$J_i(\sigma)$ – множина завдань, що виконується пристроєм i;

$T_i(\sigma) = \sum\limits_{j \in J_i(\sigma)} k_i p_j$ – тривалість зайнятості пристрою i;

$\Delta_i(\sigma) = \max\{0;\ T_i(\sigma) - C^*\} = \max\left\{0;\ \sum\limits_{j \in J_i(\sigma)} k_i p_j - C^*\right\}$ – *виступ* пристрою i;

$R_i(\sigma) = \max\{0;\ C^* - T_i(\sigma)\} = \max\left\{0;\ C^* - \sum\limits_{j \in J_i(\sigma)} k_i p_j\right\}$ – *резерв* пристрою i

$(i=\overline{1,m})$.

Позначимо через $T_i^{/}(\sigma) = \dfrac{T_i(\sigma)}{k_i}$ *зведену тривалість зайнятості пристрою* i $(i=\overline{1,m})$ в розкладі σ – час зайнятості пристрою, розрахований у еталонних

тривалостях виконання завдань. Відповідні зведені величини: $\Delta_i'(\sigma)=\max\{0;T_i'(\sigma)-c_i^*\};\ i=\overline{1,m};\ R_i'(\sigma)=\max\{0;c_i^*-T_i'(\sigma)\},\ i=\overline{1,m}.$

Визначимо: $I_\Delta(\sigma)$ – множина пристроїв, для яких $\Delta_i(\sigma)>0$; $I_R(\sigma)$ – множина пристроїв, для яких $R_i(\sigma)>0$; $I_0(\sigma)$ – множина пристроїв, для яких $\Delta_i(\sigma)=R_i(\sigma)=0$.

З урахуванням того, що величина d є константою (оскільки не залежить від розкладу) критерій оптимізації вихідної задачі (максимізація моменту r запуску пристроїв) $r=d-(C^*+\max\limits_{1\le i\le m}\Delta_i(\sigma))\to\max$ є еквівалентним критерію мінімізації загального часу виконання усіх завдань:

$$C^*+\max_{1\le i\le m}\Delta_i(\sigma)\to\min.$$

У свою чергу, з урахуванням того, що $C^*=const$, критерій спрощується і зводиться до мінімізації максимального виступу:

$$\max_{1\le i\le m}\Delta_i(\sigma)\to\min.$$

Рис. 2.6 ілюструє взаємозв’язок величин, що входять до вказаних критеріїв.

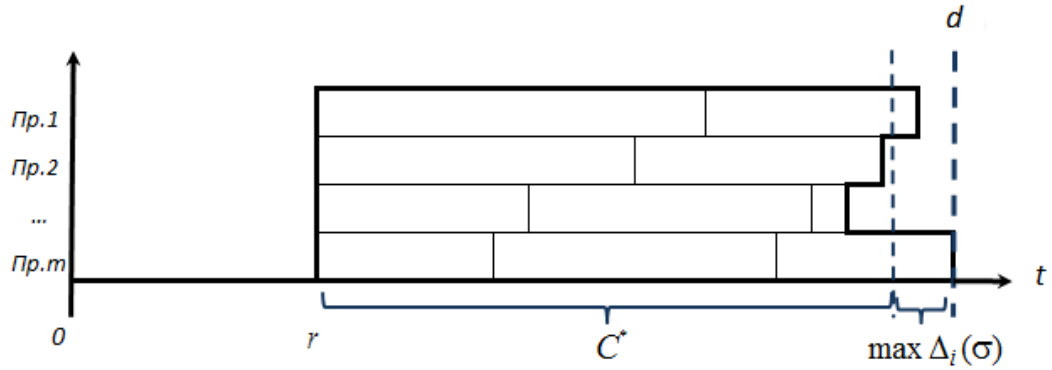


Рис. 2.6. Взаємозв’язок величин

Для введених величин має місце наступне:

$$\sum_{j\in J_i(\sigma)}k_i p_j=C^*-R_i(\sigma)+\Delta_i(\sigma),\ i=\overline{1,m}. \qquad (2.15)$$

Розділимо i–те рівняння в (2.15) на k_i ($k_i\neq 0$):

$$\sum_{j\in J_i(\sigma)}p_j=\frac{C^*}{k_i}-\frac{R_i(\sigma)}{k_i}+\frac{\Delta_i(\sigma)}{k_i},\ i=\overline{1,m},$$

$$\sum_{j\in J_i(\sigma)} p_j = c_i^* - R_i^{/}(\sigma) + \Delta_i^{/}(\sigma),\ i=\overline{1,m}.$$

Застосовуючи методологію побудови ПДС-алгоритмів [114], згідно з якою поліноміальна складова алгоритму породжується логіко-аналітичними умовами, виконання яких гарантує оптимальність отриманого рішення, визначимо ДУО розкладів.

2.2.3 Достатні умови оптимальності розкладів

Як було вказано раніше, «ідеальним» розкладом є рівномірний розклад, тобто розклад, у якого $\sum_{i=1}^{m}\Delta_i(\sigma)=\sum_{i=1}^{m}R_i(\sigma)=0$. Рівномірний розклад можливо отримати тільки тоді, коли C^* та усі $c_i^*=\dfrac{C^*}{k_i}$ є цілими числами (інакше не можливі рівності $\sum_{j\in J_i} p_j = c_i^*$, $i=\overline{1,m}$ – ліві частини рівностей за припущенням є сумами цілих величин).

Отже, ми визначили таку *ДУО розкладу*: якщо в розкладі $\sum_{j\in J_i} p_j = c_i^*$,

$(\sum_{i=1}^{m}\Delta_i^{/}(\sigma)=\sum_{i=1}^{m}R_i^{/}(\sigma)=0)$, $i=\overline{1,m}$, то поточний розклад є оптимальним.

Якщо виконується умова:

$$\exists i \mid c_i^* \notin Z, \tag{2.16}$$

(серед величин c_i^* є не цілі), то принципово неможливо побудувати рівномірний розклад. Визначимо ДУО для цього випадку.

Позначимо:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m}\left\lfloor \frac{C^*}{k_i} \right\rfloor = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m}\lfloor c_i^* \rfloor.$$

Враховуючи (2.13) маємо, що $\delta \geq 0$ і є ціле. За умови (2.16) маємо також, що $\delta > 0$ і є ціле. Визначимо, який вигляд матиме «ідеальний» розклад у цьому випадку. Іншими словами, визначимо *контур* оптимального розкладу.

Визначення. Розклад, у якому на пристрої призначені не усі завдання із множини J, будемо називати *неповним*.

Представимо, що ми розподілили між пристроями деяку кількість завдань (нехай вони складають множину $\overline{J} \subset J$) і отримали неповний розклад $\overline{\sigma}$, у якому:

$$\sum_{j \in \overline{J_i}} t_{ij} = k_i \left\lfloor c_i^* \right\rfloor, \ \ i = \overline{1,m}, \tag{2.17}$$

або

$$\sum_{j \in \overline{J_i}} p_j = \left\lfloor c_i^* \right\rfloor, \ i = \overline{1,m}, \tag{2.18}$$

тут $\overline{J_i}$ – множина завдань, які в отриманому неповному розкладі $\overline{\sigma}$ виконуються пристроєм i). Зазначимо, що залишились недорозподіленими завдання, сумарна тривалість виконання яких становить:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \sum_{j \in \overline{J_i}} p_j \ . \tag{2.19}$$

З урахуванням (2.18), рівняння (2.19) має вигляд:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \left\lfloor c_i^* \right\rfloor .$$

Припустимо, що залишилось розподілити δ завдань, еталонна тривалість кожного з яких дорівнює 1. Отже, перед нами постає така задача: розподілити одиничні завдання в кількості δ між пристроями так, щоб максимальний з виступів повного розкладу був мінімальним. В результаті ми отримаємо розклад, у якому робота в об'ємі $\sum_{j=1}^{n} p_j$ розподілена найкращим чином (ідеальний розклад для сукупності завдань з заданою сумою тривалостей їх виконання). Позначимо:

$$e_i = c_i^* - \sum_{j \in \overline{J_i}} p_j \ , \ i = \overline{1,m}, \tag{2.20}$$

де e_i – резерв пристрою i у неповному розкладі $\overline{\sigma}$, в якому недорозподілено δ одиничних завдань.

З урахуванням (2.18) маємо:

$$e_i = c_i^* - \lfloor c_i^* \rfloor,\ i = \overline{1,m}. \qquad (2.21)$$

З (2.21) слідує, що $\forall\ e_i \geq 0$.

На рис. 2.7 відображено контур найкращого неповного розкладу $\overline{\sigma}$, у якому на пристрої призначені завдання в об'ємі $\sum_{i=1}^{m} \lfloor c_i^* \rfloor$ (в еталонних тривалостях).

Просумуємо рівняння (2.21) по i: $\sum_{i=1}^{m} e_i = \sum_{i=1}^{m} c_i^* - \sum_{i=1}^{m} \lfloor c_i^* \rfloor$, з урахування (2.18) маємо:

$$\sum_{i=1}^{m} e_i = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor = \delta.$$

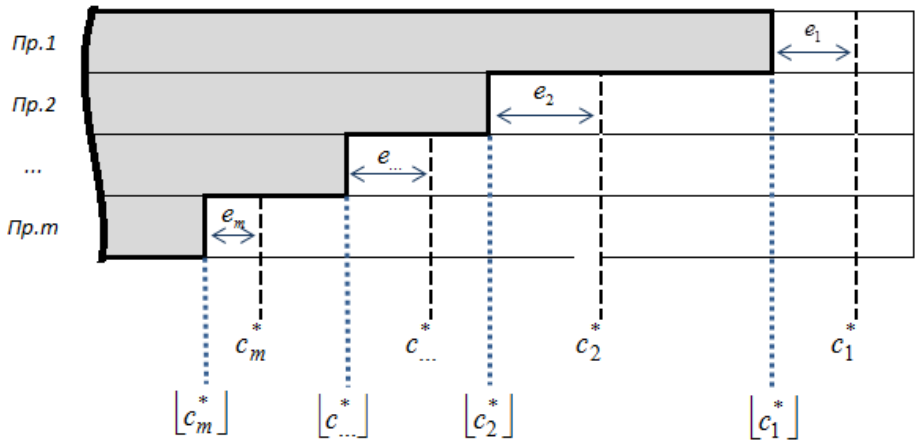


Рис. 2.7. Контур найкращого розкладу, у якому на пристрої призначені завдання в об'ємі $\sum_{i=1}^{m} \lfloor c_i^* \rfloor$ (в еталонних тривалостях)

Таким чином, можна сформулювати наступну ***допоміжну оптимізаційну задачу***: необхідно δ завдань одиничної довжини розподілити між m пристроями, за умов, що пристрій i має резерв e_i, $i = \overline{1,m}$ і $\sum_{i=1}^{m} e_i = \delta$ з метою мінімізації максимального з виступів (з урахуванням продуктивності пристроїв). Математична модель цієї задачі має наступний вигляд.

Змінні: x_i – кількість «одиничних» еталонних завдань, що повинні бути призначені на пристрій i, $i = \overline{1,m}$.

Обмеження – кількість «одиничних» еталонних завдань становить δ:

$$\sum_{i=1}^{m} x_i = \delta;\ \ x_i \geq 0,\ \ \text{ціле},\ i = \overline{1,m}. \tag{2.22}$$

Цільова функція: мінімізація максимального з виступів (з урахуванням продуктивностей пристроїв):

$$\max_i \{k_i (x_i - e_i)\} \to \min. \tag{2.23}$$

Алгоритм розв'язання допоміжної оптимізаційної задачі

Дано: δ; e_i, $i = \overline{1,m}$ ($\sum_{i=1}^{m} e_i = \delta$); k_i, $i = \overline{1,m}$; $T_i = C^* - e_i k_i$ (тривалість виконання завдань пристроєм i в неповному ідеальному розкладі $\overline{\sigma}$), $i = \overline{1,m}$.

КРОК 1 Ініціалізація: $x_i = 0$, $i = \overline{1,m}$.

КРОК 2 Поки $\delta \geq 1$ (поки є нерозподілені одиничні завдання)

2.1 Знайти:

$$\min \{T_1 + k_1;\ T_2 + k_2;\ldots;\ T_m + k_m\}, \tag{2.24}$$

($T_i + k_i$ – тривалість виконання усіх завдань пристроєм i за умови, що на цей пристрій буде призначено поточне одиничне завдання). Нехай мінімум в (2.24) відповідає пристрою $i = q$.

2.2 Призначити на пристрій q одиничне завдання: $x_q = x_q + 1, \delta = \delta - 1$.

2.3 Перерахувати тривалість виконання усіх завдань пристроєм q: $T_q = T_q + k_q$. **КІНЕЦЬ АЛГОРИТМУ**

Отже, отримали такі достатні умови оптимальності.

Достатня умови оптимальності 1

Якщо $C^* = \sum_{j=1}^{n} \frac{1}{\sum_{i=1}^{m} \frac{1}{k_i p_j}}$ – ціле та усі $c_i^* = \frac{C^*}{k_i}$ також є цілими числами – для заданого набору завдань можливо побудувати рівномірний розклад.

Якщо у розкладі σ виконується:

$$T_i'(\sigma) = \sum_{j \in J_i} p_j = c_i^*,\ i = \overline{1,m},\ (T_i(\sigma) = k_i \sum_{j \in J_i} p_j = k_i c_i^* = C^*,\ i = \overline{1,m})$$

або

$$\sum_{i=1}^{m}\Delta_i(\sigma)=\sum_{i=1}^{m}R_i(\sigma)=0,$$

то це означає, що поточний розклад є рівномірним (оптимальним).

Достатня умови оптимальності 2

Якщо $\delta=\sum_{j=1}^{n}p_j-\sum_{i=1}^{m}\lfloor c_i^*\rfloor>0$, $(\exists i\mid c_i^*=\frac{C^*}{k_i}\notin Z)$, то у цьому випадку отримати рівномірний розклад не можливо.

Нехай x_i^*, $i=\overline{1,m}$ – оптимальний розв'язок допоміжної оптимізаційної задачі (2.23)-(2.24), $T_i^*=C^*+k_i\left(x_i^*-e_i\right)$ – тривалість виконання завдань в «ідеальному» розкладі. Тоді розклад σ, в якому:

$T_i^{/}(\sigma)=\lfloor c_i^*\rfloor+x_i^*$, $i=\overline{1,m}$ або $T_i(\sigma)=T_i^*$, $i=\overline{1,m}$, є оптимальним.

З урахуванням ДУО розкладів визначимо множину операцій обміну, яка дозволить послідовно покращувати значення критерію.

2.2.4 Розробка множини операцій обміну

Отже, отримали контур оптимального розкладу для заданої сумарної довжини завдань множини J, у якого i-й пристрій завершує свої завдання у момент: C^*, якщо $\delta=0$; T_i^*, якщо $\delta>0$ (рис. 2.8).

Введемо позначення:

$$Z_i(\sigma)=\max\left\{0;\,T_i(\sigma)-T_i^*\right\}=\max\left\{0;\,\sum_{j\in J_i(\sigma)}k_ip_j-T_i^*\right\}$$ (аналог величини $\Delta_i(\sigma)$);

$$E_i(\sigma)=\max\left\{0;\,T_i^*-T_i(\sigma)\right\}=\max\left\{0;\,T_i^*-\sum_{j\in J_i(\sigma)}k_ip_j\right\}$$ (аналог величини $R_i(\sigma)$);

$I_Z(\sigma)$ – множина таких пристроїв, для яких $Z_i(\sigma)>0$;

$I_E(\sigma)$ – множина таких пристроїв, для яких $E_i(\sigma)>0$.

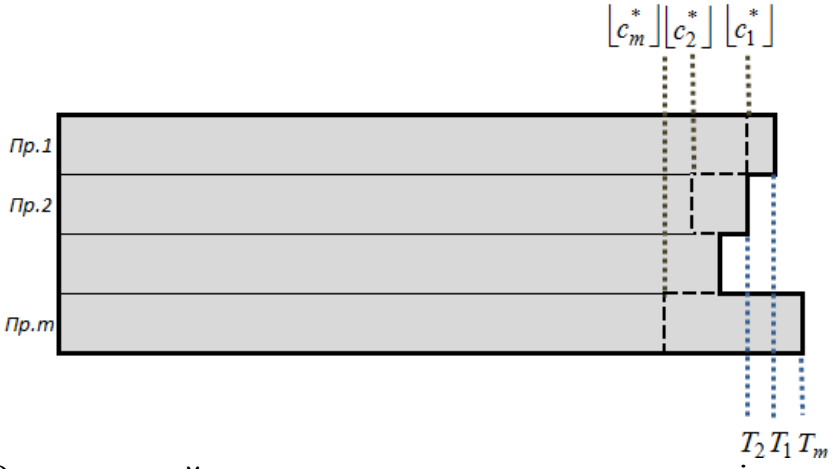


Рис. 2.8. Оптимальний за другою умовою оптимальності контур розкладу

Для покращення розкладу необхідно направити зусилля на зменшення величини $\max\limits_i Z_i(\sigma)$. Для цього пропонується використовувати обмін завдань між двома пристроями: перший пристрій h з множини $I_Z(\sigma)$, а другий – пристрій s з множин $I_E(\sigma)$, $I_Z(\sigma)$, $I_0(\sigma)$, ($s \neq h$): коли деяка підмножина завдань з пристрою h (позначимо її як $K_h(\sigma)$, $K_h(\sigma) \subseteq J_h(\sigma)$) обмінюється з деякою підмножиною завдань з пристрою s (позначимо цю підмножину як $L_s(\sigma)$, $L_s(\sigma) \subseteq J_s(\sigma)$). Позначимо через θ різницю між сумами еталонних тривалостей завдань, які приймають участь в обміні:

$$\theta = \sum_{j \in K_h(\sigma)} p_j - \sum_{j \in L_s(\sigma)} p_j \ .$$

Нехай σ^1 – розклад, який отримано після виконання операції обміну.

Для того, щоб перестановка призвела до покращення (тобто $\sum\limits_{i=1}^{m} Z_i(\sigma) - \sum\limits_{i=1}^{m} Z_i(\sigma^1) > 0$), необхідно, щоб $\theta > 0$.

У роботі розроблена множина операцій обміну, які в залежності від умов виконання та наслідків розділено на чотири типи: $\boldsymbol{A}k$, $\boldsymbol{Б}k$, $\boldsymbol{B}k$ та $\boldsymbol{\Gamma}k$. Операції обміну усіх типів, в свою чергу, можна розділити на підтипи в залежності від кількості завдань, які приймають участь в обміні (потужностей множин $K_h(\sigma)$ та $L_s(\sigma)$). Таблиця 2.3 містить узагальнену характеристику та властивості операцій

обміну всіх типів. В додатку А.2 на рис. А.2.1 – А.2.3 наведена графічна ілюстрація деяких типів обміну.

В результаті виконання перестановок типу $\boldsymbol{A}k$, $\boldsymbol{Б}k$ та $\boldsymbol{B}k$ для обраної пари пристроїв $h-s$ отримуємо таке зменшення значення максимального з виступів цих пристроїв в двох розкладах:

$$Z_h(\sigma)-\max\{Z_h(\sigma^1), Z_s(\sigma^1)\}=\min\{Z_h(\sigma),\ \theta\cdot k_h,\ Z_h(\sigma)+E_s(\sigma)-\theta\cdot k_s\}.$$

В результаті виконання операцій обміну типу $\boldsymbol{Г}k$ для обраної пари пристроїв $h-s$ отримуємо наступне зменшення значення максимального з виступів цих пристроїв в двох розкладах:

$$Z_h(\sigma)-\max\{Z_h(\sigma^1),\ Z_s(\sigma^1)\}=\min\{\theta\cdot k_h;\ Z_h(\sigma)-Z_s(\sigma)-\theta\cdot k_s\}.$$

Розроблені операції обміну покладені в основу поліноміальної складової ПДС-алгоритму розв'язання задачі.

Таблиця 2.3

Властивості операцій обміну

Тип операції обміну	**Умови, при яких виконується обмін**	**Характеристики результуючого розкладу σ^1**
$\boldsymbol{A}k$	$\theta>0$, $\theta k_h \le Z_h(\sigma)$, $\theta k_s \le E_s(\sigma)$	$Z_h(\sigma^1)=Z_h(\sigma)-\theta k_h$, $E_s(\sigma^1)=E_s(\sigma)-\theta k_s$
$\boldsymbol{Б}k$	$\theta>0$, $\theta k_s \le Z_h(\sigma)$, $\theta k_h > E_s(\sigma)$	$Z_h(\sigma^1)=Z_h(\sigma)-\theta k_s$, $Z_s(\sigma^1)=\theta k_h-E_s(\sigma)$
$\boldsymbol{B}k$	$\theta>0$, $\theta k_s > Z_h(\sigma)$, $\theta k_h \ge E_s(\sigma)$	$Z_h(\sigma^1)=\theta k_s-Z_h(\sigma)$, $Z_s(\sigma^1)=\theta k_h-E_s(\sigma)$
$\boldsymbol{Г}k$	$\theta>0$, $\theta k_s < Z_h(\sigma)$, $\theta k_h < Z_h(\sigma)-Z_s(\sigma)$	$Z_h(\sigma^1)=Z_h(\sigma)-\theta k_s$, $Z_s(\sigma^1)=Z_s(\sigma)+\theta k_h$

2.2.5 ПДС-алгоритм розв'язання задачі

На основі ДУО та розробленої множини операцій обміну побудований алгоритм визначення максимально пізнього моменту початку виконання завдань

паралельними пристроями різної продуктивності із загальним жорстким директивним строком в допустимому розкладі.

Опис алгоритму

КРОК 0 Визначити δ. Розв'язати за необхідності допоміжну оптимізаційну задачу. Визначити оптимальний контур (достатні умови оптимальності).

КРОК 1 Побудувати початковий розклад σ^0.

КРОК 2 $\sigma = \sigma^0$.

КРОК 3 Перевірити виконання достатніх умов оптимальності

ЯКЩО для δ виконується одна з достатніх умов оптимальності

ТО перейти на **КРОК 6** (σ – оптимальний розклад).

КРОК 4 Визначити пристрій h, якому відповідає максимальне значення виступу: $Z_h(\sigma) = \max\limits_{i \in I_Z(\sigma)} Z_i(\sigma)$.

КРОК 5 Виконати для пристрою h, перебираючи усі пристрої $s \in I_E(\sigma) \cup I_0(\sigma) \cup I_Z(\sigma)$, операцію обміну типу $\boldsymbol{A}k$, $\boldsymbol{Б}k$, $\boldsymbol{B}k$ або $\boldsymbol{\Gamma}k$ (отримали розклад σ^1)

ЯКЩО таких обмінів не знайшлось,

ТО

5.1 Для пристрою h перебираючи пристрої $s \in I_0(\sigma) \cup I_Z(\sigma)$ виконати операцію обміну типу $\boldsymbol{\Gamma}k$ (отримали розклад σ^1).

5.2 ЯКЩО таких операцій обму не знайшлось,

ТО перейти на **КРОК 6**,

ІНАКШЕ $\sigma = \sigma^1$, перейти на **КРОК 3.**

ІНАКШЕ $\sigma = \sigma^1$, перейти на **КРОК 3.**

КРОК 6 Визначити максимально пізній момент запуску завдань на виконання: $r_{\max} = r(\sigma) = d - \max\limits_i T_i(\sigma)$. **КІНЕЦЬ АЛГОРИТМУ**

Складність алгоритму складає $O\left(n^2 W\right)$, де $W = \sum\limits_{i=1}^{n} p_i$. Обґрунтування складності алгоритму подібне тому, як це описано у п. 2.1.5.

Якщо для результуючого розкладу σ виконується одна з ДУО, то цей розклад є оптимальним. В іншому випадку значення цільової функції розкладу σ відрізняється від її оптимального значення на величину не більше ніж $\max_i Z_i(\sigma)$. Тобто ця величина дає верхню межу відхилення значення критерія отриманого розв'язку від оптимального значення.

Приклад застосування алгоритму вирішення задачі наведено у додатку Б.

2.3 Експериментальне дослідження ефективності алгоритмів

2.3.1 Класифікація задач

Ефективність алгоритму (ступінь досягнення оптимального розв'язку) залежить від набору вхідних даних задачі. Інтуїтивно зрозуміло, що швидкість і ефективність алгоритму для системи «на кожному з пристроїв виконується досить велика кількість завдань, тривалості яких не дуже сильно відрізняються одна від іншої» і «на кожному з пристроїв виконується невелика кількість завдань, тривалості яких сильно відрізняються одна від іншої» будуть різними.

Отже, для проведення дослідження алгоритму, порівняння його ефективності за різних початкових умов, пропонується виділити наступні ознаки, за якими задачі можна класифікувати:

- тривалість завдань;
- «ступінь розсіювання» довжин завдань;
- «ступінь розсіювання» коефіцієнтів продуктивності пристроїв.

2.3.2 Тривалість завдань

Вважатимемо, що середній час роботи пристроїв (величина C^*) заданий. І нехай $\overline{p}$ – середнє значення довжини завдання. В залежності від співвідношення величин $\frac{C^*}{\overline{p}}$ індивідуальні задачі (задачі з конкретно визначеними значеннями $m,\ n,\ d,\ p_i, i = \overline{1,n}$) будуть відноситися до однієї із категорій:

- категорія S – системи із "короткими" завданнями; до цієї категорії

будемо відносити системи, у яких $\frac{C^*}{\overline{p}} \approx 50$;

– категорія M – системи із "середніми" завданнями; до цієї категорії будемо відносити системи, у яких $\frac{C^*}{\overline{p}} \approx 20$;

– категорія L – системи із "довгими" завданнями; до цієї категорії будемо відносити системи, у яких $\frac{C^*}{\overline{p}} \approx 5$.

2.3.3 Ступінь розсіювання довжин завдань

У залежності від категорії, до якої належить система за середнім значенням довжини завдання, маємо значення $\overline{p}$. Але довжини задач в одній системі можуть бути як близькими один до одного (отже, і до $\overline{p}$), так і значно відрізнятися. За це відповідатиме такий параметр, як Q (визначається як функція від $\overline{p}$). Він характеризує, наскільки великою буде розбіжність згенерованих довжин завдань відносно їх середнього значення. В залежності від величини коефіцієнта розсіювання Q встановлюється значення стандартного відхилення: $\sigma = \overline{p}Q$ і, відповідно, значення дисперсії, початкові задачі (системи) будуть відноситися до однієї із категорій:

– категорія S – системи із малим розсіюванням; до цієї категорії будемо відносити системи, у яких $Q = 0{,}05$;

– категорія M – системи із середнім розсіюванням; до цієї категорії будемо відносити системи, у яких $Q = 0{,}25$;

– категорія L – системи із великим розсіюванням, до цієї категорії будемо відносити системи, у яких $Q = 0{,}45$.

2.3.4 Ступінь розсіювання коефіцієнтів продуктивності

Початкові задачі (системи) будуть відноситися до однієї із категорій:

– категорія S – системи із "малим" розсіюванням коефіцієнтів продуктивності; до цієї категорії будемо відносити системи, у яких $1/2 \le k \le 2$;

– категорія M – системи із "середнім" розсіюванням; до цієї категорії будемо відносити системи, у яких $1/5 \le k \le 5$;

– категорія L – системи із "великим" розсіюванням, до цієї категорії будемо відносити системи, у яких $1/10 \le k \le 10$.

Для класифікації набору вхідних даних задачі надалі будемо використовувати запис *P/Q/H*:

– P характеризує тривалість завдань: $P \in \{S, M, L\}$;

– Q характеризує ступінь розсіювання довжин завдань: $Q \in \{S, M, L\}$;

– H характеризує ступінь розсіювання коефіцієнтів продуктивності пристроїв: $H \in \{S, M, L\}$.

Задача з підрозділу 2.1 є частковим випадком задачі з підрозділу 2.2, тому при експериментальних дослідженнях третій параметр (параметр H) не враховувався.

2.3.5 Результати експериментів

Для перевірки ефективності розроблених алгоритмів була проведена серія експериментів.

Наведемо результати дослідження алгоритму розв’язання задачі визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним строком паралельними пристроями різної продуктивності.

Дослідження проводились за такою схемою:

- спочатку послідовно генерувались множини задач класів *P/Q/S,* де $P \in \{S, M, L\}$, $Q \in \{S, M, L\}$ і тривалості задач – рівномірно розподілені випадкові величині;

- потім послідовно генерувались множини задач класів *P/Q/S,* де $P \in \{S, M, L\}$, $Q \in \{S, M, L\}$ і тривалості задач – нормально розподілені випадкові величині;

- для кожного з розглянутих випадків застосовувалися три різні стратегії пошуку операцій обміну: пошук першого допустимого «простого» обміну (далі: *simple/first*), пошук найкращого з можливих «простих» обмінів (далі: *simple/best*), пошук найкращого з можливих «складних» обмінів (далі: *complex/best*);
- і головне – генерувалися такі задачі, для яких відоме оптимальне значення критерію (тобто, генерувалась така множина завдань, для якої принципово можливо побудувати рівномірний розклад).

Примітка. «Простими» операціями обміну вважаються операції, у яких з кожного пристрою задіяне не більше одного завдання. До «простих» відносяться обміни «1-1» та «1-0». «Складними» операціями обміну вважаються операції, що допускають наявність двох завдань, які переставляються з одного пристрою на інший. При використанні «складних» обмінів алгоритмом можуть обиратися як обміни «2-2», «2-1» та «2-0», так і «прості» «1-1» та «1-0».

У таблиці 2.4 наведені результати експериментів за умов рівномірного розподілу тривалостей завдань.

Таблиця 2.4

Результати експериментів над алгоритмом розв'язання задачі (тривалості завдань розподілені рівномірно)

P	**Q**	**H**	**Стратегія пошуку обміну**	**Середній час розв. однієї задачі (с)**	**Середня частка відхил. від оптимуму**	**Частка знайдених оптимумів**	**Середня кількість виконаних обмінів**
1	2	3	4	5	6	7	8
S	*S*	*M*	*simple/first*	0.05	4.53E-07	0.052	112.52
S	*S*	*M*	*simple/best*	1.94	4.63E-07	0.06	25.11
S	*S*	*M*	*complex/best*	1224.78	0	1	18.42
S	*M*	*M*	*simple/first*	0.06	1.48E-06	0	62.92
S	*M*	*M*	*simple/best*	0.75	1.38E-06	0	23.41
S	*M*	*M*	*complex/best*	129.22	0	1	20.72
S	*L*	*M*	*simple/first*	0.06	2.22E-06	0	42.07
S	*L*	*M*	*simple/best*	0.47	2.30E-06	0.001	21.53
S	*L*	*M*	*complex/best*	165.37	0	1	22.17

Продовження табл. 2.4

1	2	3	4	5	6	7	8
M	*S*	*M*	*simple/first*	0.04	3.71E-03	0	100.09
M	*S*	*M*	*simple/best*	0.71	4.36E-06	0	31.35
M	*S*	*M*	*complex/best*	62.71	1.35E-07	0.538	29.77
M	*M*	*M*	*simple/first*	0.03	1.96E-05	0	40.54
M	*M*	*M*	*simple/best*	0.19	1.95E-05	0	19.79
M	*L*	*M*	*simple/first*	0.02	3.19E-05	0	29.08
M	*M*	*M*	*complex/best*	11.33	4.12E-07	0.176	26.47
M	*L*	*M*	*simple/best*	0.13	3.26E-05	0	17.41
M	*L*	*M*	*complex/best*	6.75	5.78E-07	0.149	26.24
L	*S*	*M*	*simple/first*	0.01	9.13E-04	0	23.60
L	*S*	*M*	*simple/best*	0.04	8.89E-04	0	14.91
L	*S*	*M*	*complex/best*	0.29	1.24E-04	0	20.07
L	*M*	*M*	*simple/first*	0.01	1.29E-03	0	20.15
L	*M*	*M*	*simple/best*	0.03	1.20E-03	0	13.07
L	*M*	*M*	*complex/best*	0.21	2.20E-04	0	19.21
L	*L*	*M*	*simple/first*	0.01	1.70E-03	0	14.69
L	*L*	*M*	*simple/best*	0.02	1.72E-03	0	10.17
L	*L*	*M*	*complex/best*	0.16	2.45E-04	0	16.76

У таблиці 2.5 наведені результати експериментів за умов нормального розподілу тривалостей завдань.

Таблиця 2.5

Результати експериментів над алгоритмом розв’язання задачі (тривалості завдань розподілені нормально)

P	**Q**	**H**	**Стратегія пошуку обміну**	**Середній час розв.однієї задачі**	**Середня частка відхил. від оптимуму**	**Частка знайдених оптимумів**	**Середня кількість виконаних обмінів**
1	2	3	4	5	6	7	8
M	*S*	*M*	*simple/first*	0.04	9.07E-04	0	102.11
M	*S*	*M*	*simple/best*	0.66	4.23E-06	0	29.71
M	*S*	*M*	*complex/best*	65.93	1.05E-07	0.710	30.29

Продовження таблиці 2.5

1	*2*	*3*	*4*	5	6	7	8
M	*M*	*M*	*simple/first*	0.03	1.91E-05	0	36.22
M	*M*	*M*	*simple/best*	0.17	1.98E-05	0	18.96
M	*M*	*M*	*complex/best*	7.76	5.16E-07	0.065	24.94
M	*L*	*M*	*simple/first*	0.02	3.20E-05	0	25.34
M	*L*	*M*	*simple/best*	0.11	3.15E-05	0	15.27
M	*L*	*M*	*complex/best*	4.29	5.16E-07	0.097	24.32

У додатку В наведено графічна ілюстрація аналізу проведення результатів експериментів.

Висновок до розділу 2

У розділі наведено результати дослідження задач визначення максимально пізнього моменту початку виконання в допустимому розкладі завдань із спільним директивним строком паралельними ідентичними пристроями та пристоями різної продуктивності.

Для задачі складання розкладів роботи паралельних ідентичних пристроїв розроблено множину операцій обміну завданнями між пристроями, на основі достатніх умов оптимальності розроблено поліноміальну складову ПДС-алгоритму вирішення задачі, що використовує цю множину операцій обміну.

При досліджені властивості задачі календарного планування виконання завдань із спільним жорстким директивним строком паралельними пристроями різної продуктивності з метою максимізації моменту запуску пристроїв за умови, що усі завдання не запізнюються, сформульована допоміжна оптимізаційна задача, за результатами якої визначені достатні умови оптимальності розкладів.

На основі ДУО розроблено поліноміальна складова ПДС-алгоритму вирішення задачі, що використовує множину операцій обміну завданнями між пристроями, які дозволяють послідовно покращувати значення критерію.

Для перевірки ефективності розроблених алгоритмів була проведена серія експериментів. Для цього запропонована класифікація індивідуальних задач складання розкладів, в основу якої покладені такі характеристики: кількість,

середня тривалість завдань та закон їх розподілу, «ступінь розсіювання» довжин завдань, «ступінь розсіювання» коефіцієнтів продуктивності пристроїв. Під час проведення експериментів на вхід алгоритму подавалися серії задач різних типів. Також при цьому були досліджені різні стратегії пошуку операцій обміну (пошук першого допустимого «простого» обміну (*simple/first*), пошук найкращого з можливих «простих» обмінів (*simple/best*), пошук найкращого з можливих «складних» обмінів (*complex/best*)).

На підставі результатів експериментів отримані такі висновки: найкращі результати демонструє стратегія *complex/best* на коротких завданнях незалежно від розсіювання (100-відсоткове попадання в оптимум); але, в той же час, стратегія *complex/best* є найповільнішою з розглянутих, тому може бути неприйнятною при потребі швидко розв’язати велику кількість задач; для задач з короткими завданнями усі стратегії пошуку операцій обміну (для всіх задач) дають результати, близькі до оптимуму (відсоток відхилення коливається від 10^{-7} до 10^{-5}; як і очікувалось, збільшення розсіювання тривалостей завдань скорочує час, необхідний для пошуку розв’язку, це особливо помітно на задачах з короткими завданнями; стратегія *simple/first* дозволяє отримувати розв’язок дуже швидко, але при цьому дає найбільший (проте все ще прийнятний) відсоток відхилення від оптимуму; щодо впливу розподілу тривалостей завдань виявлено, що у порівнянні з рівномірним розподілом, стратегія *complex/best* для нормального розподілу працює швидше при великих дисперсіях тривалостей, але при цьому частка знайдених оптимальних розв’язків є дещо менша (в середньому 15% проти 8%).

3 СКЛАДАННЯ ДОПУСТИМИХ РОЗКЛАДІВ ВИКОНАННЯ ЗАВДАНЬ ПАРАЛЕЛЬНИМИ ПРИСТРОЯМИ З МЕТОЮ МІНІМІЗАЦІЇ МАКСИМУМУ ВІДХИЛЕННЯ МОМЕНТІВ ЗАВЕРШЕННЯ ЗАВДАНЬ ПРИСТРОЯМИ ВІД ЗАДАНОГО СТРОКУ

Розглядаються дві задачі складання розкладів виконання завдань паралельними пристроями (рис. 3.1):

– однакової продуктивності з метою мінімізації максимуму відхилення моментів завершення завдань пристроями від загального директивного строку;

– різної продуктивності з метою побудови розкладу з максимально рівномірним розподілом завдань між пристроями (розкладу, у якому досягає мінімуму максимальне з відхилень моментів завершення виконання завдань пристроями від ідеального значення моменту завершення завдань).

Задачі близькі за змістом: у першій задачі величина, відносно якої вимірюється відхилення є заданою (загальний директивний строк завдань), а в другій ця величина є розрахунковою (представляє собою ідеальне значення моменту завершення виконання усіх завдань).

Друга задача є більш узагальненою. Якщо в ній пристрої будуть ідентичними і метою буде мінімізація максимуму відхилення моментів завершення завдань пристроями від заданого загального директивного строку, то отримаємо першу задачу. Для цих задач не відомий поліноміальний алгоритм розв’язання.

Така задача вже досліджувалася, про що свідчать наступні роботи [123, 128, 133, 149, 173-177, 183 та інші]. Для її вирішення були запропоновані різні методи, а саме: алгоритм зозулі [123], поліноміальний алгоритм [128], різні евристичні алгоритми [133, 173], метод гілок та меж [174] та інші.

На відміну від цих робіт, для задач, що розглядаються, запропоновано підхід (який базується на ПДС-методології), який полягає в тому, що для цих задач визначені достатні умови оптимальності, які покладені в основу алгоритмів розв’язання. Це дозволяє: по-перше, скоротити кількість ітерацій знаходження

розв'язку, по-друге, при виконанні достатніх умов оптимальності можна однозначно стверджувати про оптимальність отриманого розв'язку та по-третє, при не виконанні цих умов визначається величина, на яку в найгіршому випадку відрізняється величина отриманого значення критерію від оптимального.

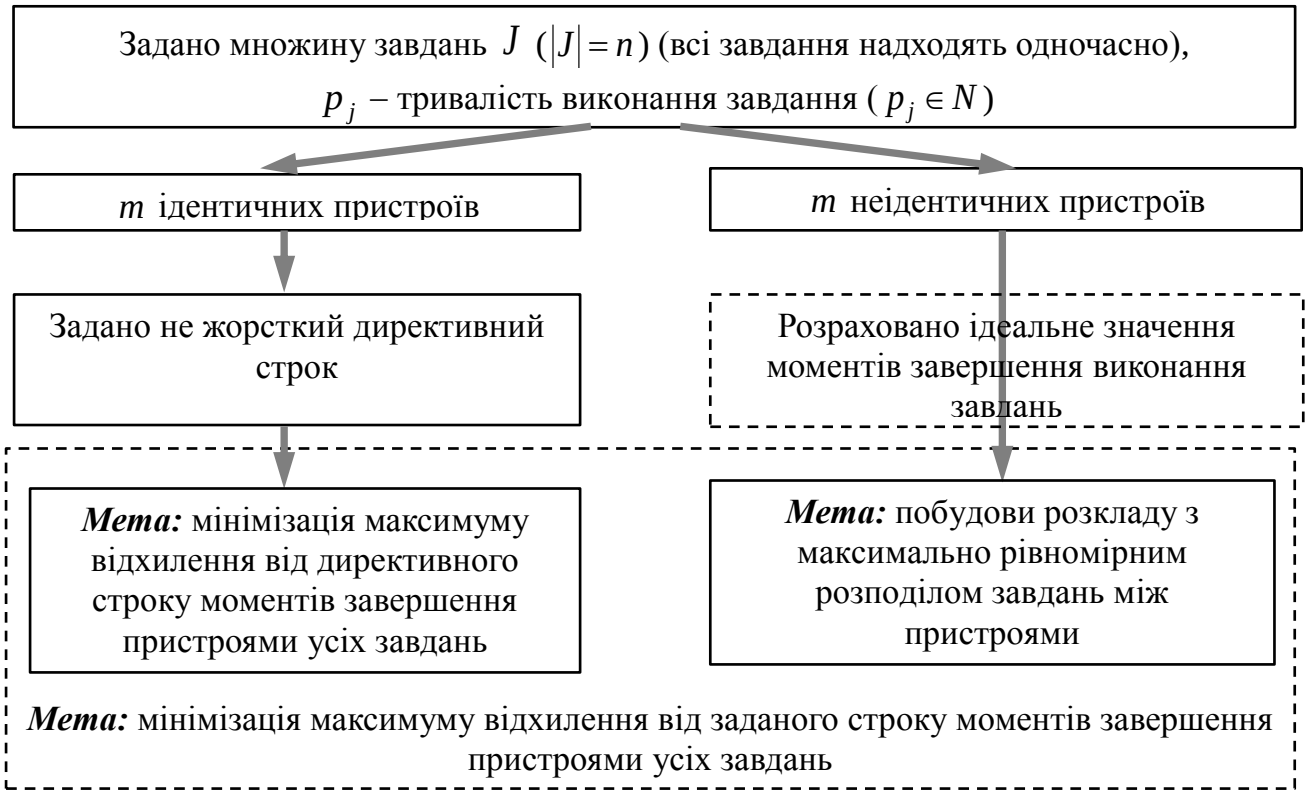


Рис. 3.1. Характеристики задач, що розглядаються у розділі 3

3.1 Задача складання розкладу виконання завдань паралельними ідентичними пристроями з метою мінімізації максимуму відхилення від директивного строку моментів завершення завдань пристроями

3.1.1 Постановка задачі

Задано множину завдань J ($|J| = n$), кількість пристроїв m, для завдання $j \in J$ відома тривалість виконання p_j. Передбачається, що всі завдання надходять одночасно і мають спільний директивний строк d ($d \in N$, $p_j \in N$, $j = \overline{1,n}$ – директивний строк та тривалість виконання завдань являються натуральними числами). Усі пристрої починають свою роботу в нульовий момент часу. Процес виконання завдань кожним з пристроїв є безперервним:

після виконання першого по порядку відразу ж починає виконуватися друге і т.д. Завдання виконуються без переривань.

Необхідно знайти розклад, в якому мінімізується максимальне відхилення від директивного строку моменту завершення пристроями всіх своїх завдань.

У роботі розглядається випадок, коли:

1) НСД значень тривалостей виконання всіх завдань ($p_j, j=\overline{1,n}$) та директивного строку (d) дорівнює одиниці (цього завжди можна досягти, розділивши ці величини на їх НСД);

2) сумарний час, виділений пристроям на виконання всіх завдань, приблизно дорівнює загальному обсягу завдань, які повинні виконати ці пристрої:

$$\sum_{j=1}^{n} p_j \approx dm \text{ та при цьому } \left|\sum_{j=1}^{n} p_j - dm\right| < m.$$

Одним із розповсюджених критичних зауважень до моделей теорії розкладів є наступне: на практиці у випадку обмеженого фонду робочого часу парку машин (пристроїв) керівництво підприємства не набирає замовлень, об'єм яких значно превищує можливості підприємства. З огляду на це, задача 2), що розглядається, є реалістичною. Результати, які були отримані для цієї задачі приведені у роботах [5, 7, 167-169].

3.1.2 Дослідження властивостей задачі

Позначимо $\delta = \sum_{j=1}^{n} p_j - dm$. Величина δ може бути від'ємною, додатньою або приймати нульове значення. Розглянемо деякий розклад σ. Введемо для нього позначення:

$C_i(\sigma)$ – момент завершення виконання всіх завдань пристроєм i, $i=\overline{1,m}$;

$\Delta_i(\sigma) = \max\{0; C_i(\sigma) - d\}$, $i=\overline{1,m}$ (виступ пристрою i);

$R_i(\sigma) = \max\{0; d - C_i(\sigma)\}$, $i=\overline{1,m}$ (резерв пристрою i);

$I_\Delta(\sigma)$ – множина пристроїв, у яких $\Delta_i(\sigma) > 0$;

$I_R(\sigma)$ – множина пристроїв, у яких $R_i(\sigma) > 0$;

$I_0(\sigma)$ – множина пристроїв, у яких $\Delta_i(\sigma) = R_i(\sigma) = 0$;

$J_i(\sigma)$ – множина завдань, які в розкладі σ виконуються пристроєм i.

З врахуванням обраних позначень критерій задачі має вигляд:

$$\max_i \{|C_i(\sigma) - d|\} \to \min \text{ або } \max_i \{R_i(\sigma); \Delta_i(\sigma)\} \to \min .$$

Критерій оцінювання розкладу можна також інтерпретувати так: необхідно знайти такий розклад, у якому пристрої навантажені максимально рівномірно.

Позначим через Ψ клас розкладів, для яких виконується:

$$\neg\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma),\ s \in I_R(\sigma),\ p_j \le R_s(\sigma) \qquad (3.1)$$

та

$$\neg\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma),\ s \in I_R(\sigma),\ p_j \le \Delta_h(\sigma), R_s(\sigma) > 0 \qquad (3.2)$$

Твердження 3.1. Оптимальний розклад належить класу Ψ.

Доведення цього твердження подібне доведенню твердження 2.1 (див. п.2.1.2).

Доведення

Якщо для розкладу σ виконується:

$$\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma),\ s \in I_R(\sigma),\ p_j \le R_s(\sigma)$$

та/або

$$\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma),\ s \in I_R(\sigma),\ p_j \le \Delta_h(\sigma), R_s(\sigma) > 0,$$

то переміщення завдання j з пристрою h на пристрій s зберігає значення критерію або призводить до його покращення. Покажемо це.

Припустимо, що σ – розклад, який не задовольняє умову (3.1). Це означає, що на деякому пристрою $h \in I_\Delta(\sigma)$ виконується завдання j, тривалість якого не перевищує резерв деякого іншого пристрою $s \in I_R(\sigma)$: $p_j \le R_s(\sigma)$. Переміщення завдання j з пристрою h на пристрій s приведе до таких наслідків:

$$\Delta_h(\sigma^1)=\begin{cases}\Delta_h(\sigma)-p_j, \text{ якщо } p_j<\Delta_h(\sigma),\\ 0, \quad \text{якщо } p_j\geq\Delta_h(\sigma),\end{cases}$$

$$\Delta_s(\sigma^1)=\Delta_s(\sigma)=0,\ R_s(\sigma^1)=R_s(\sigma)-p_j,$$

$$R_h(\sigma^1)=\begin{cases}0, \quad \text{якщо } p_j\leq\Delta_h(\sigma),\\ p_j-\Delta_h(\sigma), \text{ якщо } p_j>\Delta_h(\sigma);\end{cases}$$

де σ^1 – розклад, отриманий в результаті перестановки.

В результаті обміну порівняно з розкладом σ міг збільшитися тільки резерв пристрою h (якщо тільки $p_j>\Delta_h(\sigma)$), але цей новоутворений резерв $R_h(\sigma^1)$ буде меньший резерва $R_s(\sigma)$.

Покажемо це:

$$R_s(\sigma)-R_h(\sigma^1)=R_s(\sigma)-(p_j-\Delta_h(\sigma))=R_s(\sigma)-p_j+\Delta_h(\sigma)>0,$$

оскільки в цьому випадку, $R_s(\sigma)-p_j\geq 0$, $\Delta_h(\sigma)>0$.

Отже, в результаті обміну отримаємо:

$$\max_{h,s}\{\Delta(\sigma^1)\}<\max_{h,s}\{\Delta_i(\sigma)\},$$

$$\max_{h,s}\{R(\sigma^1)\}<\max_{h,s}\{R_i(\sigma)\},$$

а значить, значення критерію в отриманому розкладі σ^1 не погіршиться:

$$\max_i\{R_i(\sigma^1);\Delta_i(\sigma^1)\}\leq\max_i\{R_i(\sigma);\Delta_i(\sigma)\}.$$

Якщо пристрій h або пристрій s мали в розкладі максимальне із відхилень від директивного терміну, то значення критерію після обміну покращиться.

Припустимо тепер, що σ – розклад, який не задовольняє умову (3.2). Це означає, що на деякому пристрої $h\in I_\Delta(\sigma)$ є завдання, тривалість якого не перевищує величину виступа цього пристрою $p_j\leq\Delta_h(\sigma)$, в той час як деякий інший пристрій $s\in I_R(\sigma)$ має резерв $R_s(\sigma)>0$. Переміщення завдання j з пристрою h на пристрій s приведе до таких наслідків.

При цьому: $\max_{h,s}\{\Delta(\sigma^1)\}<\max_{h,s}\{\Delta_i(\sigma)\}$, $\max_{h,s}\{R(\sigma^1)\}<\max_{h,s}\{R_i(\sigma)\}$,

а значить $\max_i\{R_i(\sigma^1);\Delta_i(\sigma^1)\}\leq\max_i\{R_i(\sigma);\Delta_i(\sigma)\}$.

Якщо пристрій h або пристрій s мали в розкладі максимальне із відхилень від директивного терміну по всім пристроям, то значення критерія після операції обміну покращиться. Таким чином, існує оптимальний розклад, який належить до класу Ψ (задовольняє умовам (3.1) та (3.2)). Що й потрібно було довести.

Наслідок 3.1. Якщо в розкладі σ пристрій $h \in I_\Delta(\sigma)$, у якого $\Delta_h(\sigma) > 0$, має максимальне з відхилень від директивного строку по всіх пристроях та при цьому $\exists\ j, s \mid j \in J_h(\sigma),\ s \in I_R(\sigma),\ p_j \le R_s(\sigma)$, то цей розклад не оптимальний.

Наслідок 3.2. Якщо в розкладі σ пристрій $s \in I_R(\sigma)$ зі значенням резерву $R_s(\sigma) > 0$ має максимальне із відхилень від директивного строку по всім пристроям і при цьому $\exists h,\ j \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma),\ p_j \le \Delta_h(\sigma)$, то цей розклад не оптимальний.

Наслідок 3.3. Виконання умов (3.1) та (3.2) звужує область пошуку – множину розкладів, серед яких шукається оптимальне.

Згідно твердження 3.1 та його наслідків пошук розкладів будемо здійснювати в класі Ψ.

3.1.3 Достатні умови оптимальності розкладів

Для розкладів $\sigma \in \Psi$ можливі такі взаємовиключні випадки.

I. $\delta = 0$. У цьому випадку достатньою умовою оптимальності розкладу є рівномірне завантаження пристроїв: $C_i(\sigma) = d$, $i = \overline{1,m}$.

II. $\delta \neq 0$. У цьому випадку ДУО розкладу є: $\Delta_i(\sigma) \in \{0,1\},\ i = \overline{1,m}$ та $R_i(\sigma) \in \{0,1\},\ i = \overline{1,m}$ (частина пристроїв завершують свою роботу до директивного строку, частина – після).

Для різних знаків величини δ необхідні умови оптимальності розкладів приймають наступний вигляд (табл. 3.1).

IIA. $\delta > 0$.

IIA.1. Усі пристрої завершують роботу не раніше директивного строку:

$$C_i(\sigma) \ge d,\ i = \overline{1,m},$$

або

$$R_i(\sigma)=0,\ i=\overline{1,m}. \tag{3.3}$$

У цьому випадку розклад, в якому:

$$\Delta_i(\sigma)\in\{0,1\}\ i=\overline{1,m}, \tag{3.4}$$

є оптимальним (при цьому кількість пристроїв з ненульовим виступом дорівнює δ, рис. 3.2а).

ІІА.2. В оптимальному розкладі можуть мати місце пристрої з ненульовим (одиничним) резервом. Для збереження оптимальності розкладу, кожен додатковий ненульовий (одиничний) резерв повинен бути компенсований ненульовим (одиничним) виступом. При виконанні умов (3.3)-(3.4) кількість пристроїв, в яких $R_i(\sigma)=\Delta_i(\sigma)=0$, складає $m-\delta$. Отже, максимально можлива кількість пристроїв з $R_i(\sigma)=1$ складає $\frac{m-\delta}{2}$, якщо різниця $m-\delta$ парна, і $\left\lfloor\frac{m-\delta}{2}\right\rfloor$, якщо ця різниця непарна (тут: $\lfloor a\rfloor$ – найбільше ціле, для якого виконується: $\lfloor a\rfloor\le a$). При цьому максимально можлива кількість пристроїв з $\Delta_i(\sigma)=1$ складає $\delta+\left\lfloor\frac{m-\delta}{2}\right\rfloor$.

Таблиця 3.1

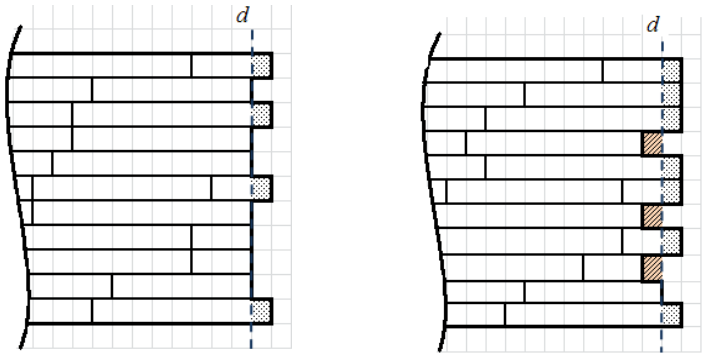
Характеристики розкладів для випадку *ІІА*

Характеристика розкладу	**Мінімальне значення**	**Максимальне значення**
Кількість пристроїв з ненульовим виступом, $\lvert I_\Delta(\sigma)\rvert$	δ	$\delta+\left\lfloor\frac{m-\delta}{2}\right\rfloor$
Кількість пристроїв з ненульовим резервом, $\lvert I_R(\sigma)\rvert$	0	$\left\lfloor\frac{m-\delta}{2}\right\rfloor$

На рис. 3.2 наведена графічна ілюстрація ДУО ***ІІА***.1 (а) та ***ІІА***.2 (б). На рис. 3.2а зображений випадок, в якому величина $\lvert I_\Delta(\sigma)\rvert$ приймає мінімально можливе значення, а на рис. 3.2б – максимально можливе (три одиничних резерви були компенсовані трьома додатковими одиничними виступами).

ІІБ: $\delta<0$.

IIБ.1. Момент завершення всіх завдань кожним пристроєм не перевищує директивного строку:



а) б)

Рис. 3.2. Графічна ілюстрація ДУО для випадків ***IIA.1*** та ***IIA.2***

$$C_i(\sigma) \le d,\ i=\overline{1,m} \quad \text{або} \quad \Delta_i(\sigma)=0,\ i=\overline{1,m}.$$

У цьому випадку розклад, в якому: $R_i(\sigma)\in\{0,1\},\ i=\overline{1,m}$ є оптимальним (при цьому кількість пристроїв з ненульовим резервом складає $|\delta|$).

IIБ.2. В оптимальному розкладі можуть бути пристрої з ненульовим (одиничним) виступом. Для збереження оптимальності розкладу, кожен додатковий ненульовий (одиничний) виступ повинен бути компенсований ненульовим (одиничним) резервом. При цьому, розклад має характеристики, які наведені у табл. 3.2.

Таблиця 3.2

Характеристики розкладів для випадку *IIБ*

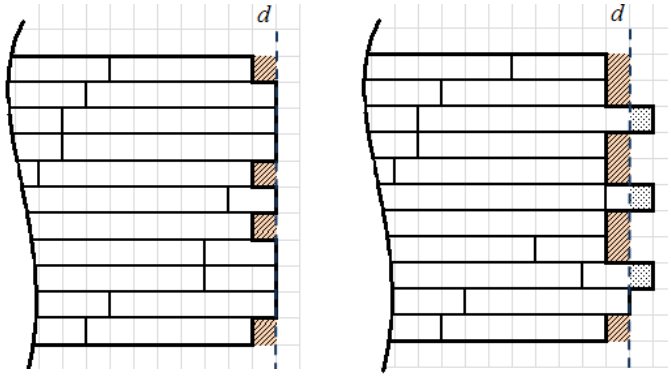
Характеристика розкладу	**Мінімальне значення**	**Максимальне значення**
Кількість пристроїв з ненульовим резервом, $\lvert I_R(\sigma)\rvert$	$\lvert\delta\rvert$	$\lvert\delta\rvert+\left\lfloor\frac{m-\lvert\delta\rvert}{2}\right\rfloor$
Кількість пристроїв з ненульовим виступом, $\lvert I_\Delta(\sigma)\rvert$	0	$\left\lfloor\frac{m-\lvert\delta\rvert}{2}\right\rfloor$

На рис. 3.3 наведена графічна ілюстрація умов оптимальності для випадків ***IIБ***.1 та ***IIБ***.2.

Продовжимо аналіз задачі.

Наслідок 3.3. Для довільного розкладу σ виконується:

$$\sum_{i=1}^{m}\Delta_i(\sigma)=\sum_{i=1}^{m}R_i(\sigma)+\delta.$$



а) б)

Рис. 3.3. Графічна ілюстрація ДУО для випадків ***ІІБ***.1 та ***ІІБ***.2

Доведення.

Час занятості пристрою i:

$$C_i(\sigma)=d-R_i(\sigma)+\Delta_i(\sigma),\ i=\overline{1,m}. \tag{3.5}$$

З іншого боку, маємо:

$$C_i(\sigma)=\sum_{j\in J_i(\sigma)}p_j. \tag{3.6}$$

З урахуванням (3.6) перепишемо (3.5)

$$\sum_{j\in J_i(\sigma)}p_j=d-R_i(\sigma)+\Delta_i(\sigma). \tag{3.7}$$

Просумуємо (3.7) по i:

$$\sum_{i=1}^{m}\sum_{j\in J_i(\sigma)}p_j=\sum_{i=1}^{m}(d-R_i(\sigma)+\Delta_i(\sigma)),$$

$$\sum_{i=1}^{n}p_j=\sum_{i=1}^{m}d-\sum_{i=1}^{m}R_i(\sigma)+\sum_{i=1}^{m}\Delta_i(\sigma),$$

$$\sum_{i=1}^{n}p_j=md-\sum_{i=1}^{m}R_i(\sigma)+\sum_{i=1}^{m}\Delta_i(\sigma),$$

$$\sum_{i=1}^{n}p_j-md=\sum_{i=1}^{m}\Delta_i(\sigma)-\sum_{i-1}^{m}R_i(\sigma),$$

$$\delta = \sum_{i=1}^{m} \Delta_i(\sigma) - \sum_{i=1}^{m} R_i(\sigma),$$

$$\sum_{i=1}^{m} \Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) + \delta.$$

Що й потрібно було довести.

3.1.4 Розробка множини операцій обміну

Оптимізація розкладу полягає в послідовному зменшенні величини $\max_i \{R_i(\sigma); \Delta_i(\sigma)\}$, чого можна досягнути за допомогою обміну завданнями між пристроями: коли деяку підмножину завдань з пристрою h (позначимо його $K_h(\sigma)$, $K_h(\sigma) \subseteq J_h(\sigma)$) міняються місцями з деякою підмножиною завдань з пристрою s (позначимо цю підмножину як $L_s(\sigma)$, $L_s(\sigma) \subseteq J_s(\sigma)$), $s \neq h$. Позначимо через θ різниця між сумами тривалостей завдань, які беруть участь у обміні:

$$\Theta = \sum_{j \in K_h(\sigma)} p_j - \sum_{j \in L_s(\sigma)} p_j .$$

Згідно твердження 3.2 існує взаємозв'язок між $\sum_{i=1}^{m} \Delta_i(\sigma)$ та $\sum_{i=1}^{m} R_i(\sigma)$: зміна сумарної величини виступів проводиться за рахунок зміни сумарної величини резервів. При цьому, в результаті операцій обміну, які застосовуються до розкладу σ, в новому розкладі σ^1 для величин $\sum_{i=1}^{m} \Delta_i(\sigma^1)$ та $\sum_{i=1}^{m} R_i(\sigma^1)$ виконується:

$$\sum_{i=1}^{m} \Delta_i(\sigma) - \sum_{i=1}^{m} \Delta_i(\sigma^1) = \sum_{i=1}^{m} R_i(\sigma) - \sum_{i=1}^{m} R_i(\sigma^1).$$

Множину операцій обміну, в залежності від їх наслідків, можна розділити на наступні типи:

– зменшення резерву (виступу) одного пристрою за рахунок зменшення виступу (резерву) іншого (тип ***А***);

– зменшення виступу з появою виступу на іншому пристрої (тип ***Б*** Δ);

– зменшення резерву з появою резерву на іншому пристрої (тип ***Б*** R);

– виключення виступу (резерву) пристрою з появою резерву (виступу) (тип ***В***);

– перерозподіл виступів (тип $\boldsymbol{\Gamma}\Delta$);

– перерозподіл резервів (тип $\boldsymbol{\Gamma}R$).

Задача, яка розглядається в цьому розділі (мінімізація максимального із резервів та виступів) за змістом близька до задачі з підрозділу 2.1 (мінімізація максимального із виступів). З урахуванням цього, операції обміну, розроблені в підрозд. 2.1, застосовані і для цієї задачі. До таких операцій обміну відносяться:

– обміни типу ***А*** (умови виконання представлені в табл. 2.1);

– обміни типу $\boldsymbol{Б}\Delta$ (в підрозд. 2.1 позначені як обміни типу ***Б***, умови виконання представлені в табл. 2.2);

– обміни типу ***В*** (умови виконання представлені в табл. 2.3);

– обміни типу $\boldsymbol{\Gamma}\Delta$ (в підрозд. 2.1 позначені як обміни типу ***Г***, умови виконання представлені в табл. 2.5).

Розглянемо обміни типу $\boldsymbol{Б}R$ та $\boldsymbol{\Gamma}R$, які направлені на зменьшення максимального із резервів.

Ціль обміну типу $\boldsymbol{Б}R$: зменшення максимального з резервів за рахунок виступу пристрою з множини I_Δ (з появою резерву на цьому пристрої). У результаті обміну цього типу може змільшитися множина I_R. Умови виконання обміну типу $\boldsymbol{Б}R$:

$$\theta > 0,$$

$$\theta > \Delta_h(\sigma), \tag{3.8}$$

$$\theta \le R_s(\sigma). \tag{3.9}$$

У результаті обміну отримуємо розклад σ^1, в якому:

$$R_h(\sigma^1) = \theta - \Delta_h(\sigma),$$

$$R_s(\sigma^1) = R_s(\sigma) - \theta, \quad h \in I_R(\sigma^1).$$

Резерв, який був максимальний в σ, зменшується на θ: $R_s(\sigma^1) = \max\limits_{i \in I_R(\sigma)} R_i(\sigma) - \theta$,

але при цьому для новоутвореного резерву виконується: $R_h(\sigma^1) < R_s(\sigma)$. Покажемо це: $R_s(\sigma) - R_h(\sigma^1) = R_s(\sigma) - (\theta - \Delta_h(\sigma)) = R_s(\sigma) + \Delta_h(\sigma) - \theta > 0$, оскільки, $R_s(\sigma) - \theta \geq 0$, $\Delta_h(\sigma) > 0$. Таким чином, розклад σ^1 не гірший, ніж розклад σ, а при відсутності альтернатив при виборі пристрою s – кращий ніж σ.

Обміни типу ***Б*** R також розділяють на підтипи в залежності від кількості завдань, які приймають участь в обміні. Детально розглянемо характеристики обміну ***1-1Б*** R, характеристики інших підтипів наведені в табл. 3.3.

Обмін підтипу ***1-1Б*** R: міняються місцями завдання j_1 з пристрою $h \in I_\Delta(\sigma)$ та завдання j_2 з пристрою $s \in I_R(\sigma)$, завдання j_1 та j_2 задовольняють наступним умовам:

$$\theta = p_{j_1} - p_{j_2} > 0, \quad \theta > \Delta_h(\sigma), \quad \theta \leq R_s(\sigma).$$

Для отриманого в результаті обміну розкладу σ^1 виконується:

$$R_h(\sigma^1) = (p_{j_1} - p_{j_2}) - \Delta_h(\sigma),$$
$$R_s(\sigma^1) = R_s(\sigma) - (p_{j_1} - p_{j_2}).$$

Приклад розкладу з виконанням обміну цього типу наведено у дод. А.3, рис. А.3.1.

Множина обмінів типу ***Б*** R призводить до того, що поточна потужність множини I_R збільшується, але при цьому величина максимального із резервів зменшується.

Таблиця 3.3

Характеристики обмінів типу *Б* R

Підтип обміну	**Пристрої та завдання, що приймають участь в обміні**		θ **(**$\theta > 0$**)**	**Умови, при яких виконується обмін**	**Характеристики результуючого розкладу** σ^1
	h	s			
1	2	3	4	5	6
1-1 Б R	$h \in I_\Delta(\sigma)$, $j_1 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_2 \in J_s(\sigma)$	$p_{j_1} - p_{j_2}$	$\theta > \Delta_h(\sigma)$, $\theta \leq R_s(\sigma)$	$R_h(\sigma^1) = \theta - \Delta_h(\sigma)$, $R_s(\sigma^1) = R_s(\sigma) - \theta$

Продовження таблиці 3.3

1	2	3	4	5	6
1-2 Б R	$h \in I_{\Delta}(\sigma)$, $j_1 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_2, j_3 \in J_s(\sigma)$	$p_{j_1} - (p_{j_2} + p_{j_3})$		
2-1 Б R	$h \in I_{\Delta}(\sigma)$, $j_1, j_2 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_3 \in J_s(\sigma)$	$(p_{j_1} + p_{j_2}) - p_{j_3}$		
2-2 Б R	$h \in I_{\Delta}(\sigma)$, $j_1, j_2 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_3, j_4 \in J_s(\sigma)$	$(p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4})$		

Ціль обмінів типу $\boldsymbol{\Gamma R}$: зменшення максимального значення резерву за рахунок перерозподілу резервів. Умови виконання обміну типу $\boldsymbol{\Gamma R}$:

$$\theta > 0,$$

$$\theta \le R_s(\sigma), \tag{3.10}$$

$$\theta < R_s(\sigma) - R_h(\sigma). \tag{3.11}$$

У результаті обміну отримуємо розклад σ^1, в якому: $R_s(\sigma^1) = R_s(\sigma) - \theta$, $R_h(\sigma^1) = R_h(\sigma) + \theta$, та при цьому для розкладу σ^1 виконується:

$$\sum_{i=1}^{m} R_i(\sigma^1) = \sum_{i=1}^{m} R_i(\sigma),$$

але $\max\{R_h(\sigma^1), R_s(\sigma^1)\} < \max\{R_h(\sigma), R_s(\sigma)\}$, тобто, розклад σ^1 не гірше ніж σ.

Обміни типу $\boldsymbol{\Gamma R}$ також розділяють на підтипи в залежності від кількості завдань, які приймають участь в обміні. Детально розглянемо характеристики обміну ***1-1Г R***, характеристики інших підтипів наведені в таблиці 3.4.

Обмін підтипу ***1-1Г R***: міняються місцями завдання j_1 з пристрою $s \in I_R(\sigma)$ та завданняа j_2 з пристрою $h \in I_R(\sigma) \cup I_0(\sigma)$, завдання j_1 та j_2 задовольняють наступним умовам:

$$\theta = p_{j_1} - p_{j_2} > 0,$$

$$p_{j_1} - p_{j_2} \le R_s(\sigma),$$

$$p_{j_1} - p_{j_2} < R_s(\sigma) - R_h(\sigma).$$

У результаті обміну:

$$R_s(\sigma^1) = R_s(\sigma) - (p_{j_1} - p_{j_2}),$$

$$R_h(\sigma^1) = R_h(\sigma) + (p_{j_1} - p_{j_2}).$$

Приклад розкладу з виконанням обміну цього типу наведено у додатку А.3, рис. А.3.2.

Для вибраної пари пристроїв $s - h$ у результаті обміну типу $\boldsymbol{\Gamma R}$ отримуємо таке зменьшення значення максимального із резервів цих пристроїв у двох розкладах:

$$R_s(\sigma) - \max\{R_s(\sigma) - \theta,\ R_h(\sigma) + \theta\}.$$

Таблиця 3.4

Характеристики обмінів типу $\boldsymbol{\Gamma R}$

Підтип обміну	**Пристрої та завдання, що приймають участь в обміні**		θ $(\theta > 0)$	**Умови, при яких виконується обмін**	**Характеристики результуючого розкладу σ^1**
	h	s			
1-1 ΓR	$h \in I_R(\sigma) \cup I_0(\sigma)$, $j_2 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_1 \in J_s(\sigma)$	$p_{j_1} - p_{j_2}$	$\theta \le R_s(\sigma)$, $\theta < R_s(\sigma) - R_h(\sigma)$	$R_s(\sigma^1) = R_s(\sigma) - \theta$, $R_h(\sigma^1) = R_h(\sigma) + \theta$
1-2 ΓR	$h \in I_R(\sigma) \cup I_0(\sigma)$, $j_2, j_3 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_1 \in J_s(\sigma)$	$p_{j_1} - (p_{j_2} + p_{j_3})$		
2-1 ΓR	$h \in I_R(\sigma) \cup I_0(\sigma)$, $j_3 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_1, j_2 \in J_s(\sigma)$	$(p_{j_1} + p_{j_2}) - p_{j_3}$		
2-2 ΓR	$h \in I_R(\sigma) \cup I_0(\sigma)$, $j_1, j_2 \in J_h(\sigma)$	$s \in I_R(\sigma)$, $j_3, j_4 \in J_s(\sigma)$	$(p_{j_3} + p_{j_4}) - (p_{j_1} + p_{j_2})$		

У табл. 3.5 розглянуті всі можливі співвідношення між величинами θ, Δ_h та R_s, а також вказані відповідні їм значення зміни часткової цільової функції.

Таблиця 3.5

Порівняння впливу операцій обміну на значення цільової функції

Умови	**Зменшення значення** $\max\limits_{i \in \{h,s\}} \Delta_i$	**Зменшення значення** $\max\limits_{i \in \{h,s\}} R_i$	**Зменшення значення** $\max\limits_{i \in \{h,s\}} \max\{\Delta_i, R_i\}$
1	2	3	4
$\theta \le \Delta_h$, $\theta \le R_s$	θ		

Продовження таблиці 3.5

1	2	3	4
$\Delta_h \le \theta \le R_s$, $\theta \le \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor$	Δ_h	θ	θ
$\Delta_h \le \theta \le R_s$, $\theta \ge \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor + 1$		$\Delta_h + R_s - \theta$	$\Delta_h + R_s - \theta$
$R_s \le \theta \le \Delta_h$, $\theta \le \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor$	θ	R_s	θ
$R_s \le \theta \le \Delta_h$, $\theta \ge \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor + 1$	$\Delta_h + R_s - \theta$		$\Delta_h + R_s - \theta$
$\theta \ge \Delta_h$, $\theta \ge R_s$	$\Delta_h + R_s - \theta$		

Розроблена множина операцій обміну покладена в основу ПДС-алгоритму розв’язання задачі, який має такі властивості: поліноміальна складова алгоритму (умови оптимальності та поліноміальний алгоритм, який їх перевіряє) одночасно є поліноміальної апроксимацією експоненційної складової ПДС-алгоритму.

3.1.5 ПДС-алгоритм розв’язання задачі

На підставі умов оптимальності та розробленої множини операцій обміну завданнями між пристроями побудований алгоритм вирішення задачі.

Схема алгоритму

КРОК 1 Побудувати початковий розклад σ^0, $\sigma = \sigma^0$.

КРОК 2 Визначити множини $I_\Delta(\sigma)$, $I_R(\sigma)$ або $I_0(\sigma)$.

КРОК 3 Перевірити виконання умов оптимальності

ЯКЩО виконується одна із умов оптимальності

ТО кінець, σ – оптимальний розклад.

КРОК 4 Визначити пристрій q, якому відповідає максимум $\max_i \{R_i(\sigma); \Delta_i(\sigma)\}$.

КРОК 5 ЯКЩО $q \in I_\Delta(\sigma)$, **ТО** перейти на **КРОК 6**

ІНАКШЕ ($q \in I_R(\sigma)$) перейти на **КРОК 7**.

КРОК 6 Виконати для пристрою $h = q$, перебираючи усі пристрої $s \in I_R(\sigma)$, обмін типу ***А, Б***Δ або ***В***.

ЯКЩО таких обмінів не знайшлось,

ТО

6.1 Для пристрою $h = q$ перебираючи усі пристрої $s \in I_0(\sigma) \cup I_\Delta(\sigma)$ виконати обмін типу ***Г***Δ.

6.2 **ЯКЩО** таких обмінів не знайшлось,

ТО кінець алгоритму,

ІНАКШЕ перейти на **КРОК 2.**

ІНАКШЕ перейти на **КРОК 2.**

КРОК 7 Для пристрою $s = q$ перебираючи всі пристрої $h \in I_\Delta(\sigma)$ виконати обмін типу ***А, Б***R або ***В***.

ЯКЩО таких обмінів не знайшлось, **ТО**

7.1 Для пристрою $s = q$ перебираючи всі пристрої $s \in I_0(\sigma) \cup I_R(\sigma)$ виконати обмін типу ***Г***R.

7.2 ЯКЩО таких обмінів не знайшлось,

ТО кінець алгоритму,

ІНАКШЕ перейти на **КРОК 2.**

ІНАКШЕ перейти на **КРОК 2.**

КІНЕЦЬ АЛГОРИТМУ

Можливі варіанти реалізації **КРОКІВ 6** та **7**:

1) знаходимо першу операцію обміну, яка покращує розклад та виконуємо її;
2) перебираємо всі допустимі операції обміну, знаходимо серед них найефективнішу та виконуємо її.

Для побудови початкового розкладу можна використовувати один з алгоритмів, наведених у підрозділі 2.1.4. Розклади, побудовані за цими алгоритмами, відносяться до класу розкладів Ψ.

Складність алгоритму становить $O(n^4 W)$, де $W = \sum_{i=1}^{n} p_i$.

Якщо для результуючого розкладу σ виконується одна з достатніх умов оптимальності, то цей розклад є оптимальним. В іншому випадку значення цільової функції розкладу σ відрізняється від її оптимального значення на величину не більше ніж $\max_i(\Delta_i(\sigma), R_i(\sigma))$ (якщо $\delta = 0$), $\max_i(\Delta_i(\sigma), R_i(\sigma)) - 1$ (якщо $\delta \neq 0$). Тобто, ці величини дають верхню межу відхилення значення критерія отриманого розв'язку від оптимального значення.

3.2 Задача складання розкладу виконання завдань паралельними пристроями різної продуктивності з метою максимально рівномірного завантаження пристроїв

Розглядається задача календарного планування виконання множини завдань паралельними пристроями різної продуктивності з метою побудови розкладу з максимально рівномірним розподілом завдань між пристроями (розкладу, в якого досягає мінімуму максимальне з відхилень моментів завершення виконання завдань пристроями від ідеального розрахункового).

3.2.1 Постановка задачі

Задано множину завдань $J = \{1,2,...,n\}$ та кількість пристроїв m. Пристрої працюють паралельно і є взаємозамінними у тому сенсі, що кожний з пристроїв може виконувати будь-яке завдання з множини J. Пристрої відрізняються один від одного продуктивністю виконання завдань, і є одноманітними (пропорційними). Тобто, можна впорядкувати пристрої за швидкістю виконання завдання і цей порядок однаковий для всіх завдань: для кожного пристрою i заданий коефіцієнт k_i (коефіцієнт продуктивності) такий, що тривалість виконання завдання j на пристрої i дорівнює $k_i p_j$. Як і в задачі з підрозділу 2.2 «еталонним» будемо називати пристрій з коефіцієнтом продуктивності $k = 1$. У цьому сенсі величина p_j є тривалістю виконання завдання j на еталонному

пристрої (далі вважатимемо, що всі p_j є цілими величинами). Передбачається, що всі завдання множини *J* надходять одночасно (у нульовий момент часу), процес обслуговування кожного завдання протікає без переривань до завершення обслуговування завдань. Необхідно побудувати розклад з максимально рівномірним завантаженням пристроїв.

За змістом ця задача є близькою до задачі з підрозділу 2.2. Їх відмінність у наступному: в задачі з підрозд. 2.2 ми, намагались мінімізувати загальний час виконання робіт; у поточній задачі ми намагатимемось якомога рівномірно навантажити пристрої.

Результати, які були отримані для цієї задачі приведені у роботах [10, 170].

3.2.2 Дослідження властивостей задачі

Визначимо C^* – як мінімально можливий час, за який усі пристрої могли б виконати усі завдання. Цей час розраховується так:

$$C^* = \frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} \frac{1}{k_i}} = \sum_{j=1}^{n} \frac{p_j}{\sum_{i=1}^{m} \frac{1}{k_i}}$$

(виведення цього виразу наведено в підрозділі 2.2).

В ідеальному випадку розклад є *рівномірним* – у ньому всі пристрої обслуговують усі завдання за час C^*.

Введемо величину $c_i^* = \frac{C^*}{k_i}$ – «ідеальну» зведену тривалість зайнятості пристрою i $(i = \overline{1,m})$ («ідеальний» момент завершення виконання завдань пристроєм i). У разі рівномірного розкладу справедливо наступне:

$$\sum_{j=1}^{n} p_j = \sum_{i=1}^{m} c_i^*, \qquad (3.11)$$

$$\sum_{j=1}^{n} p_j = \sum_{i=1}^{m} \frac{C^*}{k_i},$$

$$\sum_{j=1}^{n} p_j = C^* \sum_{i=1}^{m} \frac{1}{k_i}$$

(сумарний час виконання всіх завдань в еталонних значеннях тривалостей дорівнює сумі ідеальних зведених тривалостей зайнятості пристроїв).

Величина C^* може бути як цілим так і не цілим числом.

Розглянемо деякий допустимий розклад σ. Позначимо в цьому розкладі:

$J_i(\sigma)$ – множину завдань, що виконується пристроєм i;

$C_i(\sigma) = \sum_{j \in J_i(\sigma)} k_i p_j$ – момент завершення виконання усіх завдань пристроєм i $(i = \overline{1,m})$;

$\Delta_i(\sigma) = \max\{0;\ C_i(\sigma) - C^*\} = \max\left\{0;\ \sum_{j \in J_i(\sigma)} k_i p_j - C^*\right\}$ – *виступ* пристрою i;

$R_i(\sigma) = \max\{0;\ C^* - C_i(\sigma)\} = \max\left\{0;\ C^* - \sum_{j \in J_i(\sigma)} k_i p_j\right\}$ – *резерв* пристрою i $(i = \overline{1,m})$.

Позначимо через $C_i'(\sigma) = \dfrac{C_i(\sigma)}{k_i}$ *зведений момент завершення виконання всіх завдань пристроєм* i $(i = \overline{1,m})$, розраховану у еталонних тривалостях виконання завдань.

Відповідні зведені величини:

$$\Delta_i'(\sigma) = \max\{0; C_i'(\sigma) - c_i^*\},\ i = \overline{1,m},$$
$$R_i'(\sigma) = \max\{0; c_i^* - C_i'(\sigma)\},\ i = \overline{1,m}.$$

Визначимо:

$I_\Delta(\sigma)$ – множина пристроїв, для яких $\Delta_i(\sigma) > 0$;

$I_R(\sigma)$ – множина пристроїв, для яких $R_i(\sigma) > 0$;

$I_0(\sigma)$ – множина пристроїв, для яких $\Delta_i(\sigma) = R_i(\sigma) = 0$.

Для введених величин має місце наступне:

$$\sum_{j\in J_i(\sigma)} k_i p_j = C^* - R_i(\sigma) + \Delta_i(\sigma),\ i=\overline{1,m}. \qquad (3.12)$$

Розділимо i–те рівняння в (3.12) на k_i ($k_i \neq 0$):

$$\sum_{j\in J_i(\sigma)} p_j = \frac{C^*}{k_i} - \frac{R_i(\sigma)}{k_i} + \frac{\Delta_i(\sigma)}{k_i},\ i=\overline{1,m};$$

$$\sum_{j\in J_i(\sigma)} p_j = c_i^* - R_i^{/}(\sigma) + \Delta_i^{/}(\sigma),\ i=\overline{1,m}.$$

Для побудови початкового розкладу застосуємо алгоритм ***A03*** з підрозд. 2.2.2.

Застосовуючи методологію побудови ПДС-алгоритмів [1], згідно з якою поліноміальна складова алгоритму породжується логіко-аналітичними умовами, виконання яких гарантує оптимальність отриманого рішення, визначимо достатні умови оптимальності розкладів.

3.2.3 Достатні умови оптимальності розкладів

З урахуванням позначень, розкладом з максимально рівномірним завантаженням будемо вважати розклад, у якому досягає мінімуму максимальне з відхилень моментів завершення пристроями всіх своїх завдань від «ідеального» часу C^* завершення усіх завдань:

$$\max_i \{R_i(\sigma); \Delta_i(\sigma)\} \to \min.$$

Оскільки, задача, що розглядається та задача із підрозд. 2.2 вирішується для складання розкладу роботи пристроїв різної продуктивності, то для визначення достатніх умов оптимальності використовується схожий підхід.

«Ідеальним» розкладом є рівномірний розклад, тобто розклад, у якого $\sum_{i=1}^{m}\Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) = 0$. Рівномірний розклад можливо отримати тільки тоді, коли C^* і усі $c_i^* = \frac{C^*}{k_i}$ є цілими числами (інакше не можливі рівності $\sum_{j\in J_i} p_j = c_i^*$, $i=\overline{1,m}$ – ліві частини рівностей за припущенням є сумами цілих величин).

Якщо виконується умова:

$$\exists i \mid c_i^* \notin Z, \tag{3.13}$$

(серед величин c_i^* є не цілі), то у цьому випадку принципово неможливо побудувати рівномірний розклад.

Позначимо:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \left\lfloor \frac{C^*}{k_i} \right\rfloor = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor.$$

Враховуючи (3.11) маємо, що $\delta \geq 0$ і є ціле.

За умови (3.13) маємо, що $\delta > 0$. Визначимо, який вигляд матиме «ідеальний» розклад у цьому випадку. Тобто, визначимо *контур* «ідеального» оптимального розкладу (моменти завершення пристроями своїх завдань в ідеальному розкладі виконання завдань в об'ємі $\sum_{j=1}^{n} p_j$).

Представимо, що ми розподілили між пристроями деяку кількість завдань (нехай вони складають множину $\overline{J} \subset J$) і отримали неповний розклад $\overline{\sigma}$, в якому:

$$\sum_{j \in \overline{J_i}} p_j = \lfloor c_i^* \rfloor, \; i = \overline{1,m}, \tag{3.14}$$

тут $\overline{J_i}$ – множина завдань, які в отриманому неповному розкладі $\overline{\sigma}$ виконуються пристроєм залишились недорозподіленими завдання, сумарна тривалість виконання яких становить:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \sum_{j \in \overline{J_i}} p_j \,. \tag{3.15}$$

З урахуванням (3.14), рівняння (3.15) має вигляд:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor. \tag{3.16}$$

Припустимо, що залишилось розподілити δ завдань, еталонна тривалість кожного з яких дорівнює 1.

Отже, перед нами постає така задача: дорозподілити ці одиничні завдання між пристроями так, щоб максимально рівномірно навантажити пристрої. У

результаті ми отримаємо розклад, в якому робота в об'ємі $\sum_{j=1}^{n} p_j$ розподілена найкращим чином відповідно обраного критерію. Позначимо:

$$e_i = c_i^* - \sum_{j \in \overline{J_i}} p_j \,,\; i = \overline{1,m}, \tag{3.17}$$

де e_i – резерв пристрою i у неповному розкладі $\overline{\sigma}$, в якому недорозподілено δ одиничних завдань.

З урахуванням (3.16) маємо:

$$e_i = c_i^* - \lfloor c_i^* \rfloor,\; i = \overline{1,m}. \tag{3.18}$$

З (3.18) слідує, що $\forall\, e_i \geq 0$. Просумуємо рівняння (3.18) по i:

$$\sum_{i=1}^{m} e_i = \sum_{i=1}^{m} c_i^* - \sum_{i=1}^{m} \lfloor c_i^* \rfloor,$$

з урахування (3.11) маємо:

$$\sum_{i=1}^{m} e_i = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor = \delta.$$

Таким чином, можна сформулювати наступну ***допоміжну оптимізаційну задачу***: необхідно δ завдань одиничної довжини розподілити між m пристроями, за умов, що пристрій i має резерв e_i, $i = \overline{1,m}$ і $\sum_{i=1}^{m} e_i = \delta$ з метою рівномірного завантаження пристроїв (з урахуванням продуктивності пристроїв).

Якщо x_i – кількість «одиничних» еталонних завдань, призначених на пристрій i, $i = \overline{1,m}$, тоді критерій задачі має такий вигляд:

$$\max_i \left\{ \left| C^* - \left(\lfloor c_i^* \rfloor + x_i \right) k_i \right| \right\} \to \min.$$

Виконаємо деякі перетворення:

$$\left| C^* - \left(\lfloor c_i^* \rfloor + x_i \right) k_i \right| = \left| c_i^* k_i - \left(\lfloor c_i^* \rfloor + x_i \right) k_i \right| = \left| c_i^* k_i - \lfloor c_i^* \rfloor k_i - x_i k_i \right| =$$

$$= \left| \left(c_i^* - \lfloor c_i^* \rfloor \right) k_i - x_i k_i \right| = \left| e_i k_i - x_i k_i \right|.$$

Таким чином критерій задачі матиме вигляд: $\max_i \left\{ \left| k_i \left(e_i - x_i \right) \right| \right\} \to \min$.

Отже, математична модель цієї задачі має наступний вигляд.

Змінні: x_i – кількість «одиничних» еталонних завдань, що повинні бути призначені на пристрій i, $i = \overline{1,m}$.

Обмеження – кількість «одиничних» еталонних завдань становить δ:

$$\sum_{i=1}^{m} x_i = \delta; \quad x_i \geq 0, \text{ цілі}, \quad i = 1,m. \tag{3.19}$$

Цільова функція: максимально рівномірне завантаження пристроїв (з урахуванням продуктивностей пристроїв):

$$\max_i \left\{ \left| k_i (e_i - x_i) \right| \right\} \to \min. \tag{3.20}$$

Алгоритм розв'язання допоміжної оптимізаційної задачі

Дано: δ; e_i $(i = 1,m)$, $\sum_{i=1}^{m} e_i = \delta$; k_i $(i = 1,m)$; $T_i = C^* - e_i k_i$ (тривалість виконання завдань пристроєм i в неповному ідеальному розкладі $\overline{\sigma}$) $(i = 1,m)$.

КРОК 1 Ініціалізація**:** $x_i = 0$, $\vartheta_i = 0$, $i = 1,m$.

КРОК 2 Поки $\delta \geq 1$ (поки є нерозподілені одиничні завдання)

2.1 Знайти варіант призначення поточного одиничного завдання:

$$\min \begin{Bmatrix} \max\left\{ k_1 \left| x_1 - e_1 + \vartheta_1 \right|; k_2 \left| - e_2 + \vartheta_2 \right|; \ldots; k_m \left| - e_m + \vartheta_m \right| \right\} \\ \max\left\{ k_1 \left| - e_1 + \vartheta_1 \right|; k_2 \left| x_2 - e_2 + \vartheta_2 \right|; \ldots; k_m \left| - e_m + \vartheta_m \right| \right\} \\ \ldots \\ \max\left\{ k_1 \left| - e_1 + \vartheta_1 \right|; k_2 \left| - e_2 + \vartheta_2 \right|; \ldots; k_m \left| x_m - e_m + \vartheta_m \right| \right\} \end{Bmatrix} \tag{3.21}$$

Нехай мінімум відповідає пристрою $i = q$.

2.2 Призначити на пристрій q одиничне завдання: $x_q = x_q + 1$, $\vartheta_q = x_q - e_q$, $\delta = \delta - 1$.

2.3 Перерахувати тривалість виконання усіх завдань пристроя q: $T_q = T_q + k_q$.

КІНЕЦЬ АЛГОРИТМУ

Отже, отримали такі достатні умови оптимальності розкладів.

Достатня умови оптимальності 1. Випадок, коли принципово можливо отримати рівномірний розклад

Якщо $C^* = \frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} \frac{1}{k_i}} = \sum_{j=1}^{n} \frac{p_j}{\sum_{i=1}^{m} \frac{1}{k_i}}$ та усі $c_i^* = \frac{C^*}{k_i}$ є цілими числами і при цьому в розкладі σ виконується: $C_i^{/}(\sigma) = \sum_{j \in J_i} p_j = c_i^*,\ i = \overline{1,m}$,

$$(C_i(\sigma) = k_i \sum_{j \in J_i} p_j = k_i c_i^* = C^*,\ i = \overline{1,m})$$

або

$$\sum_{i=1}^{m} \Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) = 0,$$

то це означає, що поточний розклад є рівномірним (оптимальним).

Достатня умови оптимальності 2. Випадок, коли принципово не можливо отримати рівномірний розклад

Якщо $\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor > 0$ $(\exists i \mid c_i^* = \frac{C^*}{k_i} \notin Z)$, то розв'язати допоміжну оптимізаційну задачу.

Нехай $x_i^*,\ i = \overline{1,m}$ – оптимальний розв'язок допоміжної оптимізаційної задачі (3.19)-(3.20), $T_i^* = C^* + k_i\left(x_i^* - e_i\right)$ – тривалість виконання завдань i-м пристроєм в «ідеальному» розкладі. Тоді розклад σ, в якому:

$C_i^{/}(\sigma) = \lfloor c_i^* \rfloor + x_i^*,\ i = \overline{1,m}$ або $C_i(\sigma) = T_i^*,\ i = \overline{1,m}$, є оптимальним.

З урахуванням достатніх умов оптимальності розкладів визначимо множину операцій обміну, яка дозволить послідовно покращувати значення критерію.

3.2.4 Розробка множини операцій обміну

Отже, отримали контур оптимального розкладу для заданої сумарної довжини завдань множини J, у якого i-й пристрій завершує свої завдання в момент: C^*, якщо $\delta = 0$; T_i^*, якщо $\delta > 0$.

Введемо позначення:

$$Z_i(\sigma) = \max\left\{0;\ C_i(\sigma) - T_i^*\right\} = \max\left\{0;\ \sum_{j \in J_i(\sigma)} k_i p_j - T_i^*\right\}$$ (аналог величини $\Delta_i(\sigma)$, але в даному випадку розраховується відхилення від «ідеальної» тривалості зайнятості пристрою T_i^* в оптимальному контурі розкладу);

$$E_i(\sigma) = \max\left\{0;\ T_i^* - C_i(\sigma)\right\} = \max\left\{0;\ T_i^* - \sum_{j \in J_i(\sigma)} k_i p_j\right\}$$ (аналог величини $R_i(\sigma)$);

$I_Z(\sigma)$ – множина таких пристроїв, для яких $Z_i(\sigma) > 0$; $I_E(\sigma)$ – множина таких пристроїв, для яких $E_i(\sigma) > 0$.

Для покращення розкладу необхідно направити зусилля на зменшення величини $\max\limits_i Z_i(\sigma)$. Для цього пропонується використовувати обмін завдань між пристроями з множин $I_Z(\sigma)$ та $I_E(\sigma)$, $I_Z(\sigma)$, $I_0(\sigma)$, ($s \neq h$): коли деяка підмножина завдань з пристрою h (позначимо її як $K_h(\sigma)$, $K_h(\sigma) \subseteq J_h(\sigma)$) обмінюється з деякою підмножиною завдань з пристрою s (позначимо цю підмножину як $L_s(\sigma)$, $L_s(\sigma) \subseteq J_s(\sigma)$). Позначимо через θ різницю між сумами еталонних тривалостей завдань, які приймають участь в обміні:

$$\Theta = \sum_{j \in K_h(\sigma)} p_j - \sum_{j \in L_s(\sigma)} p_j \ .$$

Нехай σ^1 – розклад, який отримано після виконання обміну. Для того, щоб обмін завданнями призвів до покращення (тобто $\sum\limits_{i=1}^{m} Z_i(\sigma) - \sum\limits_{i=1}^{m} Z_i(\sigma^1) > 0$), необхідно, щоб $\theta > 0$.

Множину операцій обмінів, у залежності від їх наслідків, можна розділити на наступні типи:

– зменьшення резерву (виступу) одного пристрою за рахунок зменшення виступу (резерву) іншого (тип $\boldsymbol{A}k$);

– зменшення виступу з появою виступу на іншому пристрої (тип $\boldsymbol{Б}\Delta k$);

– зменшення резерву з появою резерву на іншому пристрої (тип $\boldsymbol{Б}Rk$);

– виключення виступу (резерву) пристрою з появою резерву (виступу) (тип $\boldsymbol{B}k$);

– перерозподіл виступів (тип $\boldsymbol{Г}\Delta k$);

– перерозподіл резервів (тип $\boldsymbol{Г}Rk$).

Задача, яка розглядається в цьому розділі (рівномірне навантаження пристроїв) за змістом близька до задачі з підрозд. 2.2 (мінімізація максимального з виступів у пристроях різної продуктивності). З урахуванням цього, операції обміну, розроблені в підрозд. 2.2 можна застосовувати для цієї задачі. До таких обмінів відносяться: $\boldsymbol{A}k$, $\boldsymbol{Б}\Delta k$ (у підрозд. 2.2.4 наведена як $\boldsymbol{Б}k$), $\boldsymbol{B}k$, $\boldsymbol{Г}\Delta k$ (у підрозд. 2.2.4 наведена як $\boldsymbol{Г}k$) умови виконання яких представлені в підрозд. 2.2.4 табл. 2.16. Розглянемо операції обміну, які раніше не були розглянуті, це обміни типу $\boldsymbol{Б}Rk$ та $\boldsymbol{Г}Rk$, які направлені на зменьшення максимального із резервів.

Таблиця 3.6 містить узагальнену характеристику та властивості обмінів типів $\boldsymbol{Б}Rk$ та $\boldsymbol{Г}Rk$.

Таблиця 3.6

Властивості операцій обміну

Тип операції обміну	**Умови, при яких виконується обмін**	**Характеристики результуючого розкладу σ^1**
$\boldsymbol{Б}Rk$	$\theta > 0$, $\theta k_s > Z_h(\sigma)$, $\theta k_h \le E_s(\sigma)$	$Z_h(\sigma^1) = \theta k_s - Z_h(\sigma)$, $Z_s(\sigma^1) = E_s(\sigma) - \theta k_h$
$\boldsymbol{Г}Rk$	$\theta > 0$, $\theta k_h \le E_s(\sigma)$, $\theta k_s < E_h(\sigma) - E_s(\sigma)$	$E_h(\sigma^1) = E_h(\sigma) - \theta k_s$, $E_s(\sigma^1) = E_s(\sigma) + \theta k_h$

Операції обмінів усіх типів, у свою чергу, можна розділити на підтипи в залежності від кількості завдань, які приймають участь в обміні (потужностей

множин $K_h(\sigma)$ та $L_s(\sigma)$).

У результаті виконання обмінів типу $\boldsymbol{Б}Rk$ для обраної пари пристроїв $h-s$ отримуємо таке зменшення значення максимального з резервів цих пристроїв у двох розкладах:

$$E_s(\sigma)-\max\{E_h(\sigma^1),E_s(\sigma^1)\}=\min\{E_s(\sigma),\ \theta\cdot k_s,\ E_s(\sigma)+Z_h(\sigma)-\theta\cdot k_h\}.$$

У результаті виконання операцій обміну типу $\boldsymbol{Г}Rk$ для обраної пари пристроїв $h-s$ отримуємо наступне зменшення значення максимального з резервів цих пристроїв у двох розкладах:

$$E_h(\sigma)-\max\{E_h(\sigma^1),\ E_s(\sigma^1)\}=\min\{\theta\cdot k_h;E_h(\sigma)-E_s(\sigma)-\theta\cdot k_s\}.$$

Результати виконання обмінів для зменшення значення максимального з виступів між обраною парою пристроїв наведені у підрозд. 2.2.4.

Розроблені операції обміну покладені в основу поліноміальної складової ПДС-алгоритму розв'язання задачі.

3.2.5 ПДС-алгоритм розв'язання задачі

На основі ДУО та розробленої множини операцій обміну побудований алгоритм максимально рівномірного завантаження паралельних пристроїв різної продуктивності в розкладі.

Опис алгоритму

КРОК 0 Визначити оптимальний контур (достатні умови оптимальності**)**

КРОК 1 Побудувати початковий розклад σ^0.

КРОК 2 $\sigma=\sigma^0$.

КРОК 3 Перевірити виконання достатніх умов оптимальності

ЯКЩО $\delta=0$ і виконується перша достатня умова оптимальності

ТО кінець (σ – оптимальний розклад)

КРОК 4 Визначити пристрій q, якому відповідає максимум $\max\limits_i\{Z_i(\sigma);E_i(\sigma)\}$.

КРОК 5 ЯКЩО $q\in I_Z(\sigma)$, **ТО** перейти на **КРОК 6**

ІНАКШЕ ($q\in I_E(\sigma)$) перейти на **КРОК 7**.

КРОК 6 Виконати для пристрою $h = q$, перебираючи усі пристрої $s \in I_E(\sigma)$, операцію обміну типу $\boldsymbol{A}k$, $\boldsymbol{Б}\Delta k$ або $\boldsymbol{B}k$.

ЯКЩО таких обмінів не знайшлось, **ТО**

6.1 Перебираючи усі пристрої $s \in I_0(\sigma) \cup I_Z(\sigma)$ для пристрою $h = q$ виконати операцію обміну типу $\boldsymbol{Г}\Delta k$.

6.2 ЯКЩО таких обмінів не знайшлось,

ТО кінець алгоритму,

ІНАКШЕ перейти на **КРОК 3.**

ІНАКШЕ перейти на **КРОК 3.**

КРОК 7 Перебираючи всі пристрої $h \in I_Z(\sigma)$ для пристрою $s = q$ виконати операцію обміну $\boldsymbol{A}k$, $\boldsymbol{Б}Rk$ або $\boldsymbol{B}k$.

ЯКЩО таких обмінів не знайшлось, **ТО**

7.1 Перебираючи всі пристрої $s \in I_0(\sigma) \cup I_E(\sigma)$ для пристрою $s = q$ виконати операцію обміну типу $\boldsymbol{Г}Rk$.

7.2 ЯКЩО таких обмінів не знайшлось,

ТО кінець алгоритму,

ІНАКШЕ перейти на **КРОК 3.**

ІНАКШЕ перейти на **КРОК 3.**

КІНЕЦЬ АЛГОРИТМУ

Можливі варіанти реалізації **КРОКІВ 6** та **7**:

1) знаходимо першу операцію обміну, яка покращує розклад та виконуємо її;
2) перебираємо всі допустимі обміни, знаходимо серед них найефективніший та виконуємо його.

Складність алгоритму складає $O(n^2 W)$, де $W = \sum_{i=1}^{n} p_i$. Обґрунтування складності алгоритму подібне тому, як це описано у підрозд. 2.1.5.

Якщо для результуючого розкладу σ виконується одна з ДУО, то цей розклад є оптимальним. В іншому випадку значення цільової функції розкладу

σ відрізняється від її оптимального значення на величину не більше ніж $\max_i(Z_i(\sigma), E_i(\sigma))$. Тобто ця величина дає верхню межу відхилення значення критерія отриманого розв'язку від оптимального значення.

3.3 Експериментальне дослідження ефективності алгоритмів

Для перевірки ефективності розроблених алгоритмів була проведена серія експериментів. Для цього використовувалася класифікація індивідуальних задач складання розкладів, описана в підрозділі 2.3. Під час проведення експериментів на вхід алгоритму подавалися серії задач різних типів. При цьому, були досліджені різні стратегії пошуку операцій обміну (див. п. 2.3.5.1).

На підставі результатів експериментів отримані такі висновки:

– найкращі результати демонструє стратегія *complex/best* на коротких завданнях незалежно від розсіювання (100-відсоткове попадання в оптимум), але, в той же час, стратегія *complex/best* є найповільнішою з розглянутих;

– для задач з короткими завданнями всі стратегії пошуку операцій обміну (для всіх задач) дають результати, близькі до оптимуму (відсоток відхилення коливається від 10^{-7} до 10^{-6}; як і очікувалось, збільшення розсіювання тривалостей завдань скорочує час, необхідний для пошуку розв'язку, це особливо помітно на задачах з короткими завданнями;

– стратегія simple/first дозволяє отримувати розв'язок дуже швидко, але при цьому дає найбільший (проте все ще прийнятний) відсоток відхилення від оптимуму (для коротких та середніх тривалостей відхилення не перевищує 10%).

Для перевірки ефективності розроблених алгоритмів була проведена серія експериментів. Для цього використовувалася класифікація індивідуальних задач складання розкладів, описана в підрозділі 2.3. Під час проведення експериментів на вхід алгоритму подавалися серії задач різних типів. При цьому, були досліджені різні стратегії пошуку операцій обміну (див. п. 2.3.5.1).

На підставі результатів експериментів отримані такі висновки:

– найкращі результати демонструє стратегія *complex/best* на коротких завданнях незалежно від розсіювання (100-відсоткове попадання в оптимум), але, в той же час, стратегія *complex/best* є найповільнішою з розглянутих;

– для задач з короткими завданнями усі стратегії пошуку операцій обміну (для всіх задач) дають результати, близькі до оптимуму (відсоток відхилення коливається від 10^{-7} до 10^{-6};

– як і очікувалось, збільшення розсіювання тривалостей завдань скорочує час, необхідний для пошуку розв'язку, це особливо помітно на задачах з короткими завданнями; стратегія *simple/first* дозволяє отримувати розв'язок дуже швидко, але при цьому дає найбільший (проте все ще прийнятний) відсоток відхилення від оптимуму (для коротких та середніх тривалостей відхилення не перевищує 10%).

Висновок до розділу 3

Розглянуто дві задачі складання розкладів виконання завдань паралельними пристроями. Одна з них, задача складання розкладу виконання завдань паралельними пристроями однакової продуктивності з метою мінімізації максимуму відхилення моментів завершення завдань пристроями від загального директивного строку, а інша – задача складання розкладу для пристроїв різної продуктивності з метою побудови розкладу з максимально рівномірним розподілом завдань між пристроями. Розглянуті задачі є близькі за змістом: у першій задачі величина, відносно якої вимірюється відхилення є заданою, а у другій ця величина є розрахунковою.

Проведене дослідження цих властивостей задач, у результаті яких сформульовані ДУО розкладів. При досліджені властивостей задач календарного планування виконання завдань пристроями різної продуктивності з метою побудови розкладу з максимально рівномірним розподілом завдань між пристроями сформульована допоміжна оптимізаційна задача, за результатами якої визначена одна з ДУО розкладів. Розроблено алгоритм вирішення допоміжної оптимізаційної задачі. З урахуванням ДУО розкладів визначено

множину операцій обміну, які дозволяють послідовно покращувати значення критерію. На основі ДУО та розробленої множини операцій обміну побудовані поліноміальні складові ПДС-алгоритмів вирішення поставлених задач.

Для перевірки ефективності розроблених алгоритмів була проведена серія експериментів. На підставі результатів експериментів отримані такі висновки: найкращі результати демонструє стратегія *complex/best* на коротких завданнях незалежно від розсіювання (100-відсоткове попадання в оптимум), але, в той же час, стратегія *complex/best* є найповільнішою з розглянутих; для задач з короткими завданнями усі стратегії пошуку операцій обміну (для усіх задач) дають результати, близькі до оптимуму (відсоток відхилення коливається від 10^{-7} до 10^{-6}; як і очікувалось, збільшення розсіювання тривалостей завдань скорочує час, необхідний для пошуку розв’язку, це особливо помітно на задачах з короткими завданнями; стратегія *simple/first* дозволяє отримувати розв’язок дуже швидко, але при цьому дає найбільший (проте все ще прийнятний) відсоток відхилення від оптимуму (для коротких та середніх тривалостей відхилення не перевищує 10%).

РОЗДІЛ 4 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОПЕРАТИВНО-КАЛЕНДАРНОГО ПЛАНУВАННЯ ДРІБНОСЕРІЙНОГО ВИРОБНИЦТВА

Для оперативно-календарного планування дрібносерійного виробництва, яке полягає в організації злагодженої та комплексної роботи всіх дільниць виробництва з виготовлення і випуску заданої номенклатури виробів у встановлених обсягах та строках при найкращому використанні всіх виробничих ресурсів розроблено інформаційну технологію оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк».

4.1 Архітектура та принципи функціонування інструментального комплексу ІТОКПДВ

З огляду на наявність потреби використання різними користувачами спільного набору даних, застосування розподіленої архітектури є очевидною потребою. При цьому, є два розповсюджені способи побудови архітектури: із використанням “товстого” клієнту та “тонкого”. “Товстий” клієнт представляє собою архітектурне рішення, при якому логіка роботи застосунку переноситься на сторону клієнта (користувача), а сервер виступає лише у ролі сховища даних. Такого роду клієнтом може виступати десктопне застосування. Перевагами у використанні якого є:

- доступ до апаратних засобів клієнта (користувача), як наслідок – відсутність потреби у великих власних серверних потужностях;
- можливість працювати без доступу до мережі Інтернет (із обмеженою функціональністю);
- збільшення кількості клієнтів, які можуть обслуговуватись сервером за момент часу;
- можливість користуватись технологіями, які доступні при роботі десктопних застосувань, як наслідок – висока швидкодія.

При цьому, існує ряд значних недоліків, таких як: зменшення безпеки (частина даних постійно зберігається на стороні клієнта); залежність від

апаратної архітектури та особливостей кожного клієнта; залежність від операційної системи та встановлених необхідних додаткових програмних продуктів; необхідність встановлення на кожному клієнті копії застосування; ускладнений процес міграції клієнтів на більш нові версії програмного продукту.

Застосування “тонкого” клієнта на основі JavaScript, HTML, CSS дає можливість вирішити даний ряд недоліків. На сьогоднішній день можливість підключення до мережі Інтернет не викликає великих складнощів, порівняно з тим, як це було раніше.

Тому застосування “тонкого” клієнта надає наступні переваги:

- можливість інтеграції зі сторонніми сервісами;
- легке масштабування за вимогою (при використанні хмарних ресурсів);
- значно вища відмовостійкість;
- простота підтримки клієнтів;
- незалежність від апаратної архітектури та операційної системи клієнтів;
- потреба у підтримці тільки однієї операційної системи, комплекту сторонніх допоміжних ресурсів та застосувань (які можуть не працювати при інших наборах параметрів);
- простота додавання нового функціоналу та керування доступами клієнтів до частин застосунку;
- швидкість розробки.

Недоліками є: потреба у підтримці різних версій браузерів, системі масштабування застосування, автоматичного розгортання нових версій.

З огляду на наведений вище перелік переваг та особливості даного програмного продукту, вибір архітектури “тонкого” клієнту є більш доцільним.

Інстументальний комплекс ІТОКПДВ розроблений із застосування “тонкого” клієнта та представляє собою трирівневу архітектуру, що складається із клієнтського середовища, сервера застосувань, а також сервера БД.

Подібна архітектура дозволяє отримати низку переваг:

– ізоляція сервера БД від клієнта шляхом розробки підсистеми уніфікованого доступу до даних, таким чином, забезпечується захист серверу від несанкціонованого доступу;

– через використання тонкого клієнта немає необхідності у відновленні ПЗ на робочих станціях користувачів (установка тільки один раз при розгортанні);

– автоматичне оновлення (за рахунок заміни модулів, що знаходяться тільки на сервері застосувань) клієнти завжди працюють з однаковими версіями застосувань;

– централізований контроль (має велике значення при створенні розроблюваної системи безпеки та контролю доступу), доступ до серверу застосувань має обмежена група адміністраторів, на відміну від «товстих» систем, установлених на кожному робочому місці, де кожен має свої можливості;

– дозволяє робити масштабування системи для збільшення продуктивності;

– дозволяє легко конфігурувати кожен із рівнів у зв’язку з їх розподіленням.

Архітектура інструментального комплексу ІТОКПДВ наведена на рис. 4.1.

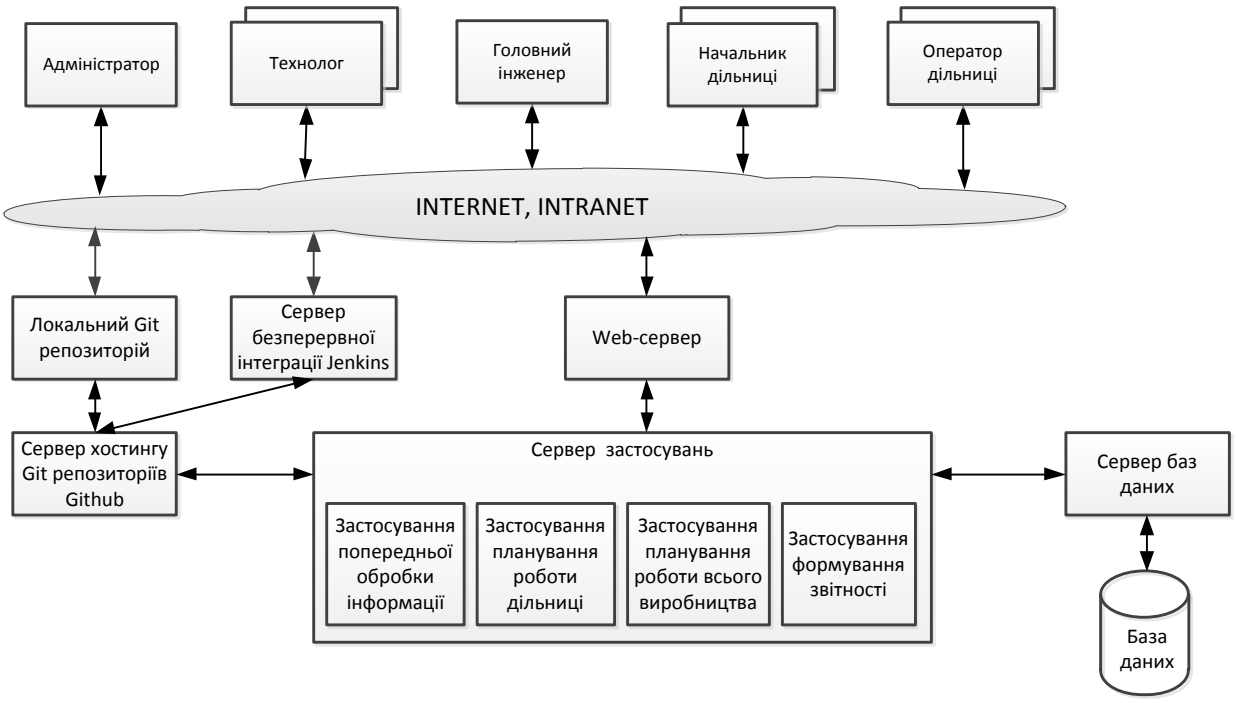


Рис. 4.1. Архітектура інструментального комплексу ІТОКПДВ

Компонентами інструментального комплексу ІТОКПДВ є:

– Web сервер. У роботі, в його якості, виступає хмарний сервіс Openshift компанії Red Hat. Даний сервіс дозволяє швидко та зручно налаштувати середовище під свій програмний продукт. Openshift дозволяє вибрати катриджі для відповідної мови програмування, які являють собою вже готові налаштовані шаблони, що забезпечує швидке розгортання програмного продукту на сервері. У залежності від вимог програмного продукту можна додавати нові катриджі до вже існуючих, наприклад катридж бази даних;

– сервер БД для роботи з даними PostgreSQL, який дозволяє швидко та зручно маніпулювати даними великого об’єму. Перевага використання даної бази в тому, що вона є компактною і зберігає всі дані в одному місці, що зменшує ризик втрати даних, оскільки її можна піддати контролю версій;

– сервер неперервної інтеграції Jenkins, який дозволяє легко здійснювати контроль релізів програмних систем. Перевагою використання цього продукту є його повна відкритість, що забезпечує стабільне оновлення версій. Jenkins є легким та зручним у налаштуванні, а також має можливість розширення за допомогою різноманітних плагінів;

– репозиторій контролю версій GitHub для зберігання програмного коду. Github є одним із найбільших репозиторіїв для спільної розробки програмного забезпечення. Він дозволяє переглядати збережений код, має функцію підсвічування синтаксису для різних мов програмування. Github працює з системою контролю версій Git;

– сервер застосувань, в основу реалізації якого покладені розроблені у роботі методи, що описані у розділах 2 та 3 цієї роботи, до його складу входять:

1) застосування попередньої обробки інформації дозволяє ввести користувачу всю необхідну інформацію для подальшого планування роботи виробництва;

2) застосування планування роботи дільниці дозволяє складати розклади роботи окремої дільниці та відслідковувати фактичне їх виконання;
3) застосування планування роботи всього підприємства дозволяє складати плани роботи підприємства та відслідковувати фактичне їх виконання;
4) застосування формування звітної документації слугує для отримання кінцевого результату у вигляді, який є зручним для кінцевого користувача.

Принцип роботи архітектури інструментального комплексу інформаційної технології від імені адміністратора представлено на рис. 4.1.

Доступ до вихідного коду сервера застосувань має адміністратор, який у свою чергу має можливість редагування поточного та внесення нового функціоналу. Для цього адміністратор робить зміни в локальному коді та зберігає їх локально в себе на комп'ютері. Після успішного тестування адміністратор завантажує зміни до хостингу коду Github.

Оновлення програмного продукту проходять за допомого серверу безперервної інтеграції Jenkins. Jenkins має зконфігуровані завдання для оновлення серверу застосування та перезапуску поточної працюючої системи. Адміністратор запускає роботу по встановленню змін, які були внесені до сервера застосувань. Після успішного виконання новий застосунок готовий до роботи. Запити до сервера застосувань надходять від клієнтів через Web-сервер. Дані, які оброблюються сервером застосувань передаються до сервера баз даних для зберігання.

При виникненні непередбачуваних ситуацій адміністратор має доступ до сервера баз даних, для відновлення інформації або усунення помилок.

Структура схеми бази даних (таблиці) контролюється за допомогою відкритого інструменту для міграції бази даних Flyway. Структура бази даних задається, як в послідовності SQL-скриптів, які містять в собі SQL-запити для створення відповідних таблиць бази даних. При розгортанні сервер застосувань

Flyway перевіряє базу даних на наявність виконання скриптів (наявність таблиць) та якщо якась таблиця відсутня створює її.

Адміністратор посилає запит до серверу неперервної інтеграції Jenkins на розгортання нової версії сервера застосувань. Сервер неперервної інтеграції робить запит до хостингу Github на копіювання останнього коду програмного забезпечення. Після копіювання коду Jenkins виконує скрипт для компіляції та упаковування вихідного коду у *.war формат. Після успішного упаковування Jenkins зупиняє сервер застосування, видаляє стару версію *.war файлу та копіює нову. Після копіювання Jenkins запускає сервер застосувань.

Якщо сервер застосувань стартував нормально та готовий до роботи він приймає запити на обробку, які надходять від клієнтів. Отриманий запит обробляється головним сервлетом, який передає запит на подальшу оброку до відповідного контролера. Контролер надсилає запит до бази даних для отримання чи на збереження інформації. Отримана інформація з бази даних далі обробляється контролером. Після успішного виконання, контролер повертає результат своєї роботи назад до клієнта у відповідному форматі.

Розглянемо детальніше архітектуру сервера застосувань. Він представляє собою багаторівневу архітектуру, наведену на рис. 4.2, яка складається з наступних рівнів: рівень представлення даних, бізнес рівень, рівень доступу до даних.

Рівень представлення даних відповідає за рендеринг кінцевої інформації для представлення її на стороні клієнта. Цей рівень реалізований за допомогою JSP-сторінок. Логіка взаємодії компонентів даного рівня реалізована за допомогою мови програмування java script із застосуванням бібліотеки JQuery.

Бізнес рівень реалізує основну логіку роботи програмного продукту. Він реалізований за допомогою контролерів Spring Framework. Рівень включає в себе чотири контролери, які відповідають за маніпуляції з даними, завантаження інформації Web-сторінок та формування звітної документації.

Рівень доступу до даних містить в собі реалізацію методів доступу до таблиць бази даних та класи об’єктно-реляційного відображення.

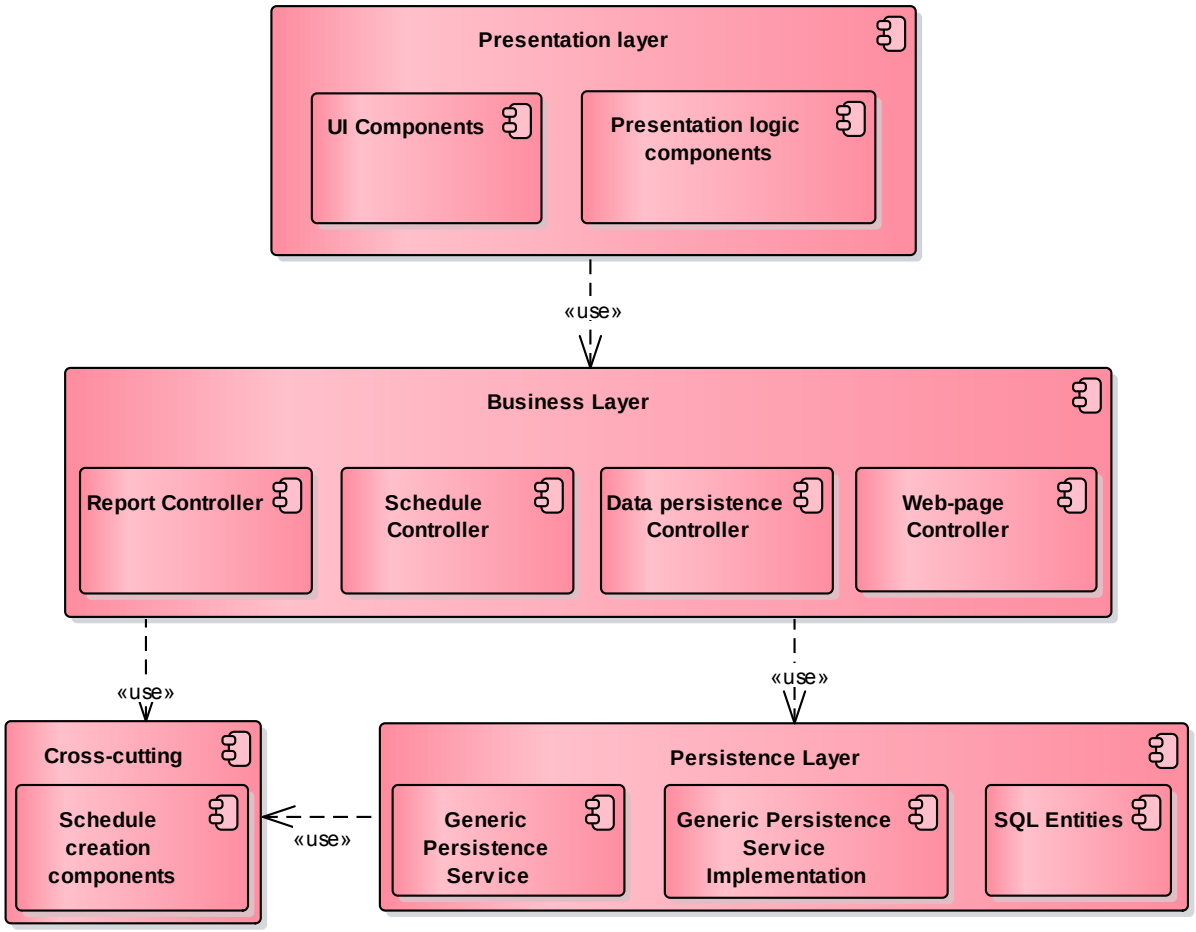


Рис. 4.2. Архітектура сервера застосувань інструментального комплексу ІТОКПДВ

Рівень доступу до даних та бізнес рівень використовують наскрізну функціональність, яка містить в собі основні алгоритмічні рішення (Cross-cuttlng).

4.2 Реалізація інструментального комплексу інформаційної технології

4.2.1 Опис пакетів та класів інструментального комплексу

Класи програмного продукту було розділено на чотири пакети (рис. 4.3) в залежності від функцій, які вони виконують:

– com.algorithm.entities – містить в собі класи-сутності, які відображають елементи графу, такі як ребра та вершини та сам граф;
– com.algorithm.scheduling – містить в собі класи, які реалізують основні алгоритми складання розкладу;

– com.controllers – містить класи-контролери, які обробляють запити, що надходять від клієнта;
– com.services – містить в собі класи, які здійснюють доступ до бази даних;
– com.sql.entities – містить в собі класи-сутності, які відображають реляційні дані з бази даних в об’єкти.

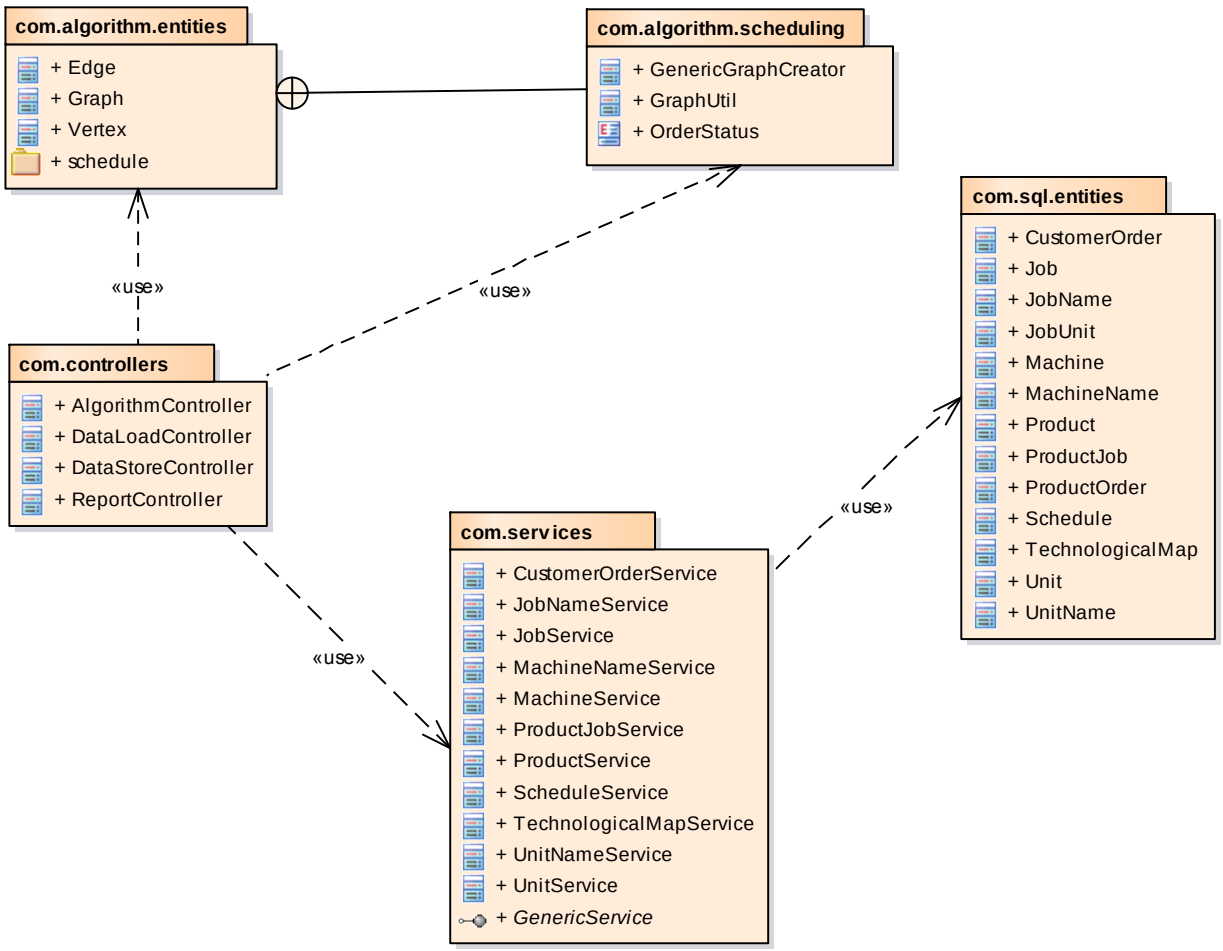


Рис. 4.3. Діаграма пакетів

Розглянемо детальніше кожен із пакетів.

Діаграма класів пакету com.algorithm.entities наведена в додатку Г на рис. Г.1.1. Даний пакет містить три класи: `Graph`, `Vertex`, `Edge`.

Клас `Graph` представляє собою програмне представлення графу. Граф задається списком вершин та містить методи для додавання нових вершин, що необхідно в процесі роботи алгоритму.

Клас `Vertex` представляє собою вершину графа. Він містить в собі список вхідних та вихідних ребер, індекс вершини в графі та посилання на виробничу дільницю.

Клас `Edge` представляє собою ребро графа. Він містить в собі інформацію про роботу та посилання на вхідну та вихідну вершини.

Діаграма класів пакету com.algorithm.scheduling наведена в додатку Г на рис. Г.1.2.

Клас `GenericGraphCreator` містить в собі методи, які реалізують алгоритм створення технологічної карти.

Клас `GraphUtil` містить в собі методи, які реалізують базові алгоритми та операції над графами, такі як пошук в ширину, визначення наявності вихідних ребер та ін.

Клас `OrderStatus` містить в собі перелік констант, які слугують для позначення процесу виконання замовлення.

Діаграма класів пакету com.controllers наведена в додатку Г на рис. Г.1.3.

Клас `AlgorithmController` містить методи обробки запитів від клієнта на створення розкладу та виведення його у відповідному форматі.

Клас `DataLoadController` містить в собі методи, які завантажують інформацію на відповідну сторінку.

Клас `DataStoreController` містить методи, які відповідають за збереження інформації до бази даних.

Клас `ReportController` містить методи, які відповідають за формування вихідної документації та діаграм.

Діаграма класів пакету com.services наведена в додатку Г на рис. Г.1.4.

Інтерфейс `GenericService` містить оголошення загальних методів для всіх сервісів, які здійснюють маніпуляції з данини бази даних.

Пакет містить в собі всі реалізації вище згаданого інтрефейсу. Реалізації інтерфейсу відповідають кожній таблиці бази даних.

Діаграма класів пакету com.sql.entities наведена в додатку Г на рис. Г.1.5.

Даний пакет містить в собі класи, які є об’єктними відображеннями реляційних сутностей бази даних.

Опис основних функцій класів програмного забезпечення з їх параметрами та значеннями, що вони повертають наведено в додатку Г.2.

4.2.2 Опис бази даних

В якості бази даних для програмного продукту використано реляційну базу даних PostgreSQL. В якості API для доступу до бази даних використано бібліотеку об’єктно-реляційного відображення Hibernate. Вибір реляційної бази даних пов’язаний з тим, вона дозволяє зручно і легко впорядковувати інформацію та зв’язки між пов’язаними даними. На рис. 4.4 представлена схема бази даних. Наведемо опис таблиць баз даних.

Таблиця CustomerOrder містить в собі всю інформацію, яка стосується замовлення користувача. Таблиця ProductOrder є допоміжною для реалізації відношення багато до багатьох. Таблиця ProductJob є допоміжною для реалізації відношення багато до багатьох. Таблиця TechnologicalMap містить в собі бінарне представлення технологічної карти виробу. Таблиця TechnologicalMap містить в собі всю необхідну інформацію про виріб. Таблиця Job містить інформацію про операцію, яку необхідно виконати для виготовлення виробу. Таблиця Machine містить інформацію про пристрій, на якому виконуються операції по виготовленню виробу. Таблиця Unit містить інформацію про виробничу дільницю, на якій виконуються операції по виготовленню виробу. Таблиця MachineName містить в собі список імен пристроїв. Таблиця UnitName містить в собі список імен виробничих дільниць. Таблиця JobName містить в собі список імен операцій. Таблиця Schedule містить в собі бінарне представлення розкладу виробництва.

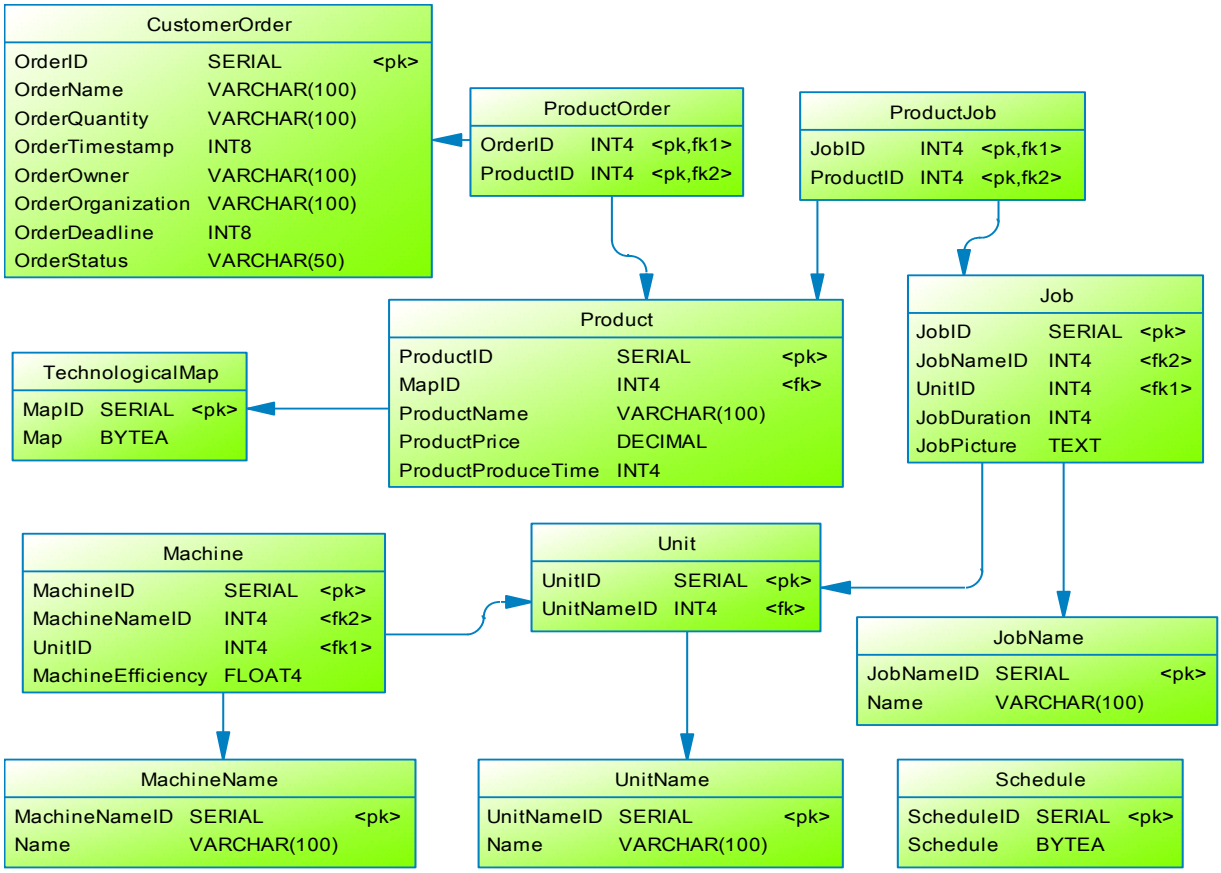


Рис. 4.4. Схема бази даних

4.3 Обґрунтування вибору технологій розробки

Інструментальний комплекс інформаційної технології ОКПДВ був розроблена з використанням наступних технологій:

– високорівнева мова програмування Java;
– реляційна база даних PostgreSQL;
– платформа Java Enterprise Edition;
– Spring MVC Framework;
– бібліотека об’єктно-реляційного відображення Hibernate.

В якості інтегрованого середовища розробки було використано Intellij Idea версії 15.0.4. Дане середовище дозволяє автоматично генерувати діаграми класів та має широкий спектр різних засобів інтеграції з іншими засобами для розробки програмного забезпечення як системи контролю версій, web-сервери, покриття коду, фреймворки та ін. [178].

Система контролю версій, яка була використана під час написання коду Git, є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні та злитті гілок.

Для розв'язання конфліктів залежностей між зовнішніми бібліотеками, використано систему автоматичної збірки проектів Maven.

Платформа Java Enterprise Edition надає API та середовище виконання для розробки та запуску корпоративного програмного забезпечення такого як мережеві сервіси, web-сервіси, розподілені системи тощо.

Java – об'єктно-орієнтована мова програмування з C подібним синтаксисом. Основною перевагою Java є те, що дана мова не залежить від архітектури та є портативною, тобто, код, який написаний на Java можна запустити на будь-якій платформі, де встановлена віртуальна машина Java без змін у початковому коді та перекомпіляції [179].

Spring Framework – універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Spring MVC – модуль Spring Framework, який базується на сервлетах та дозволяє створювати web прикладні програми і REST-сервіси. Його архітектура поділяє систему на три частини: модель даних, вигляд даних та керування. Дана архітектура застосовується для відокремлення даних від інтерфейсу користувача, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [180].

Бібліотека Hibernate для мови програмування Java призначена для вирішення задач об'єктно-реляційного відображення. Hibernate є вільним програмним забезпечення та має відкритий вихідний код. Бібліотека дозволяє значно спростити написання коду взаємодії з базою даних, оскільки звільняє розробника від написання рутинних завдань із програмування. Hibernate автоматично виконує обробку результуючого набору даних та конвертацію об'єктів, максимально полегшує перенесення коду для використання з іншою

базою даних тощо. Hibernate дає можливість організації відношень між класами «один до багатьох» та «багато до багатьох» [181].

PostgreSQL – це об'єктно-реляційна система керування базами даних (СКБД). PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати дану СКБД та впроваджувати у неї найновіші досягнення [182].

PostgreSQL використовується для реалізації великих систем, таких як: система аналізу фінансових даних, пакет моніторингу функціональності потоків, база даних відстеження астероїдів, система медичної інформації, кілька географічних систем. PostgreSQL також використовується як навчальний інструмент в кількох університетах.

PostgreSQL пропонує чотири типу індексів: B-tree, Hash, GiST і GIN. Кожен тип індексу має свій алгоритм реалізації, що дозволяє істотно збільшити швидкодію, якщо для певного виду даних вибрати певний тип індексу. PostgreSQL має вбудований у ядро повнотекстовий пошук, який дозволяє створювати запити до текстових документів, де і самі запити та порядок видачі мають гнучкі налаштування залежно від конкретних потреб.

PostgreSQL є дуже надійною СКБД. Згідно з результатами автоматизованого дослідження різного ПЗ на предмет помилок, у вихідному коді PostgreSQL було знайдено 20 проблемних місць на 775 000 рядків вихідного коду (в середньому, одна помилка на 39 000 рядків коду) [182].

Flyway представляє собою інструмент міграції бази даних з відкритим вихідним кодом. Він є зручних та легким у конфігуруванні. Flyway базується на 6 основних командах: Migrate, Clean, Info, Validate, Baseline and Repair.

Міграції можна описувати в SQL (синтаксис конкретної бази даних, наприклад, PL/SQL, T-SQL, PostgreSQL тощо) або Java.

4.4 Розробка алгоритмічного забезпечення ІТОКПДВ

Розглянемо алгоритмічне забезпечення для реалізації одних з основних процесів інформаційної технології ОКПДВ, а саме розробку та формування

технологічної карти виробу, об'ємного календарного плану виробництва, план завантаження вузлів (пристроїв) завданнями. Для реалізації останніх двох далі у роботі буде наведено побудову узагальненої технологічної карти виробництва. Результати, які були отримані для цієї задачі наведені у роботах [4, 8, 9, 171, 172].

4.4.1 Загальна математична постановка задачі

Є обслуговуюча система, що складається з множини U, $U = u$ виробничих дільниць. На кожній дільниці розміщений один або декілька пристроїв (одиниць обладнання), що можуть працювати паралельно. Всі дільниці є унікальними в сенсі того, що кожна з них виконує один і тільки один вид завдань (технологічних операцій обробки), які не можуть бути виконані на жодній іншій дільниці.

Дільниця y ($y = \overline{1,u}$) характеризується кількістю пристроїв m_y та вектором коефіцієнтів продуктивності пристроїв $M_y = \{k_{yl}\},\ y = \overline{1,u},\ l = \overline{1,m_y}$.

Системі потрібно виготовити множину виробів P, $|P| = n$. Для кожного виробу $p_i \in P$ відомий директивний строк його виготовлення d_i, $i = \overline{1,n}$.

Для виготовлення кожного виробу $p_i \in P$, $i = \overline{1,n}$, необхідно виконати множину завдань, для яких задано відношення передування. Відношення безпосереднього передування завдань, пов'язаних з виготовленням виробу p_i, задається орієнтованим ациклічним антитранзитивним графом $G_i = \langle V_i, E_i \rangle$, у якому вершини відповідають завданням, а наявність направленого ребра з вершини з індексом j_1 до вершини з індексом j_2 означає, що робота j_2 може бути розпочата лише після завершення роботи j_1 ($j_1, j_2 \in \{1,2,\ldots,g_i\}$, де $g_i = |V_i|$ – кількість завдань, що відповідають виробу i). Вимога щодо властивості антитранзитивності накладається на граф G_i, тому що він представляє відношення *безпосереднього* передування завдань: наявність дуги між

вершинами, що відповідають завданням j_1 та j_3, коли існує таке завдання j_2, що $(v_{ij_1}, v_{ij_2}) \in E_i$ і $(v_{ij_2}, v_{ij_3}) \in E_i$, є надлишковою.

Для кожного завдання (ij), $i = \overline{1,n}$, $j = \overline{1,g_i}$, відомі час її виконання t_{ij} на «еталонному» пристрої та індекс $y_{ij} \in \{1, 2, \ldots, u\}$ виробничого вузла, на якому це завдання може бути виконане. «Еталонним» будемо називати пристрій з коефіцієнтом продуктивності $k = 1$. Фактичний час виконання завдання (ij) на пристрої l вузла з індексом y_{ij} становить $\frac{t_{ij}}{k_{y_{ij}l}}$, $l = \overline{1, m_{y_{ij}}}$.

Ациклічний антитранзитивний граф $G_i = \langle V_i, E_i \rangle$ в сукупності зі значеннями часу виконання t_{ij} завдань та індексами y_{ij} виробничих вузлів ($j = \overline{1,g_i}$) утворює *технологічну карту* виробу i.

Необхідно скласти розклад виготовлення виробів, при якому усі завдання не запізнюються (жоден з директивних строків не порушується).

Процес розв'язання задачі складається з двох етапів:

Етап 1. Формування узагальненого графа проходження завдань.

1.1. Створення проміжних графів G_i^U на основі графів G_i, $i = \overline{1,n}$.

1.2. Послідовне об'єднання проміжних графів в узагальнений граф G.

Етап 2. Складання розкладу виконання завдань вузлами на основі узагальненого графа.

4.4.2 Задача формування узагальненої технологічної карти проходження завдань по виробничих дільницях

Описана вище задача може бути представлена як багатоетапна мережева задача календарного планування. Весь процес виготовлення вихідної множини виробів можна представити у вигляді ациклічного графа $G = \langle V, E \rangle$.

По суті граф G є деякою формою об'єднання всіх графів G_i, $i = \overline{1,n}$. Граф G має такі особливості: його вершини відповідають дільницям $u_y \in U$, $y = \overline{1,u}$ а

ребра відображають просування усіх виробів через систему в процесі їх виготовлення – іншими словами, ребра відповідають завданням з виготовлення кожного з виробів, які потрібно виготовити.

Пристрої, які фізично існують, виконують завдання по різних виробах у відповідності до відношення передування G_i, $i=\overline{1,n}$. Отже, у графі $G=\langle V,E\rangle$ одній і тій самій виробничій дільниці $u_y \in U$ в загальному випадку може відповідати декілька вершин з множини V. Кожному ребру поставлено у відповідність виріб та конкретне завдання з виготовлення цього виробу, при цьому якщо в графі G залишити тільки ребра, асоційовані з виробом i, та інцидентні з ними вершини, то в результаті отримаємо граф G_i.

У загальному випадку, пару вершин графа G можуть з'єднувати декілька ребер, асоційованих з різними виробами. Далі граф G будемо називати узагальненим графом проходження завдань, що представляє *узагальнену технологічну карту*.

4.4.2.1 Приклад побудови узагальненої технологічної карти

Наведемо приклад побудови узагальненої технологічної карти (далі – *узагальнений граф*).

Приклад 4.1. Нехай обслуговуюча система складається з трьох виробничих вузлів, умовні позначення яких представлені на рис. 4.5.

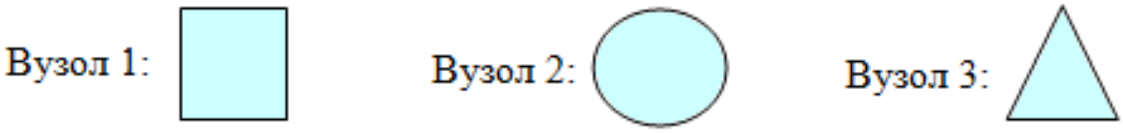


Рис. 4.5. Умовні позначення вузлів обслуговуючої системи

Ця система повинна виготовити три вироби: A, B, C, відношення безпосереднього передування завдань яких представлені на рис. 4.6–4.8.

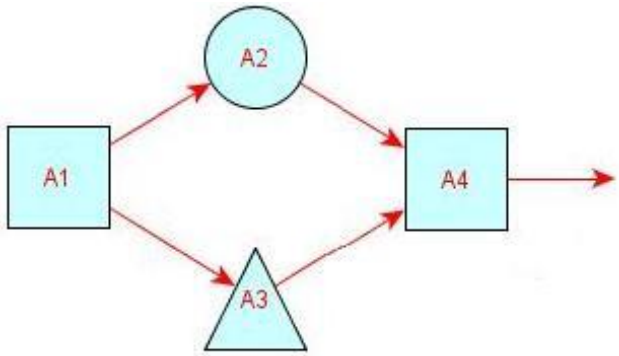


Рис. 4.6. Відношення передування завдань виробу A (граф G_A)

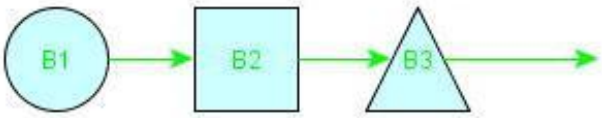


Рис. 4.7. Відношення передування завдань виробу B (граф G_B)

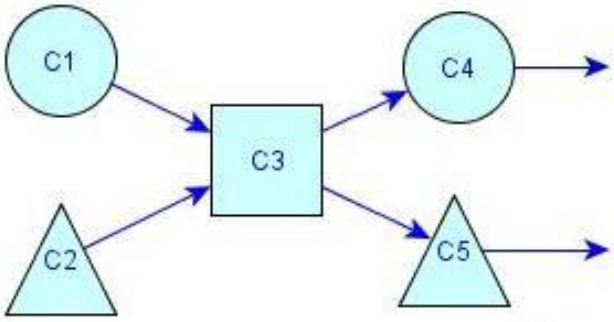


Рис. 4.8. Відношення передування завдань виробу C (граф G_C)

У графах G_i, $i = A, B, C$, представлених на рис. 4.6–4.8, вершини позначають завдання (позначення $A4$ розуміється як «четверте завдання з виготовлення виробу A») та вказують на те, яким вузлом системи кожне завдання повинно бути виконане; ребра цих графів по суті являють собою обмеження, що можна прочитати як «завдання, до вершини якої входить ребро, не може бути розпочате раніше, ніж закінчиться виконання завдання, з вершини якої це ребро виходить».

Узагальнений граф, на відміну від графів G_i, має іншу структуру. В ньому кожна вершина представляє собою вузол обслуговуючої системи, на вхід якого поступає множина завдань, що повинні бути розподілені за пристроями цього вузла. Ребра відповідають конкретним завданням, що поступають на вхід вузлів,

причому завдання, що поступають на вхід вузлів «ззовні», без втрати загальності також називатимемо ребрами такого графа. Коли на вхід одного вузла узагальненого графа поступає декілька ребер, які представляють завдання по одному й тому ж виробу, перед входом у вузол вони з'єднуються, оскільки для поточного вузла представляють одне вхідне завдання.

Таку структуру має узагальнений граф, і він отримується шляхом послідовного об'єднання множини спеціальних графів G_i^U, $i=\overline{1,n}$, які мають таку ж структуру. Далі називатимемо графи G_i^U проміжними. Проміжні графи є результатом приведення графів G_i до структури, описаної вище. Графи G_i^U, що відповідають графам G_i, $i=A,B,C$ (див. рис. 4.6–4.8), наведено на рис. 4.9–4.11.

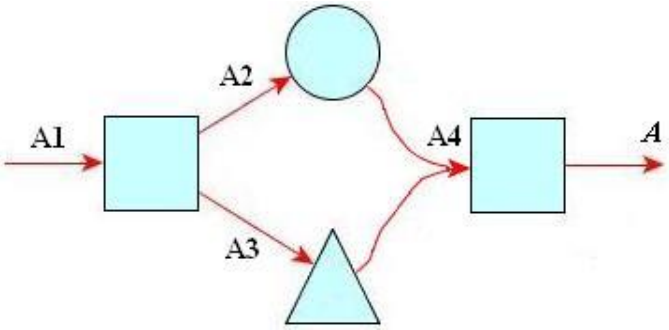


Рис. 4.9. Проміжний граф G_A^U

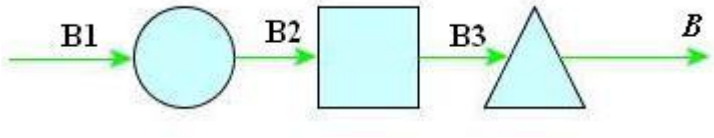


Рис. 4.10. Проміжний граф G_B^U

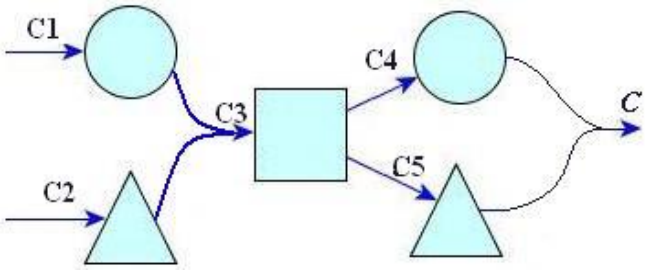


Рис. 4.11. Проміжний граф G_C^U

На рис. 4.9–4.11 стрілки, відмічені літерами A, B та C без номерів завдань, позначають готові вироби, що отримуються на виході системи, і ребрами графів не вважаються.

Узагальнений граф G отримується внаслідок послідовного з'єднання всіх графів G_i^U, $i=\overline{1,n}$, при умові збереження ациклічності результуючого графа. Для прикладу, що розглядається, узагальнений граф представлений на рис. 4.12.

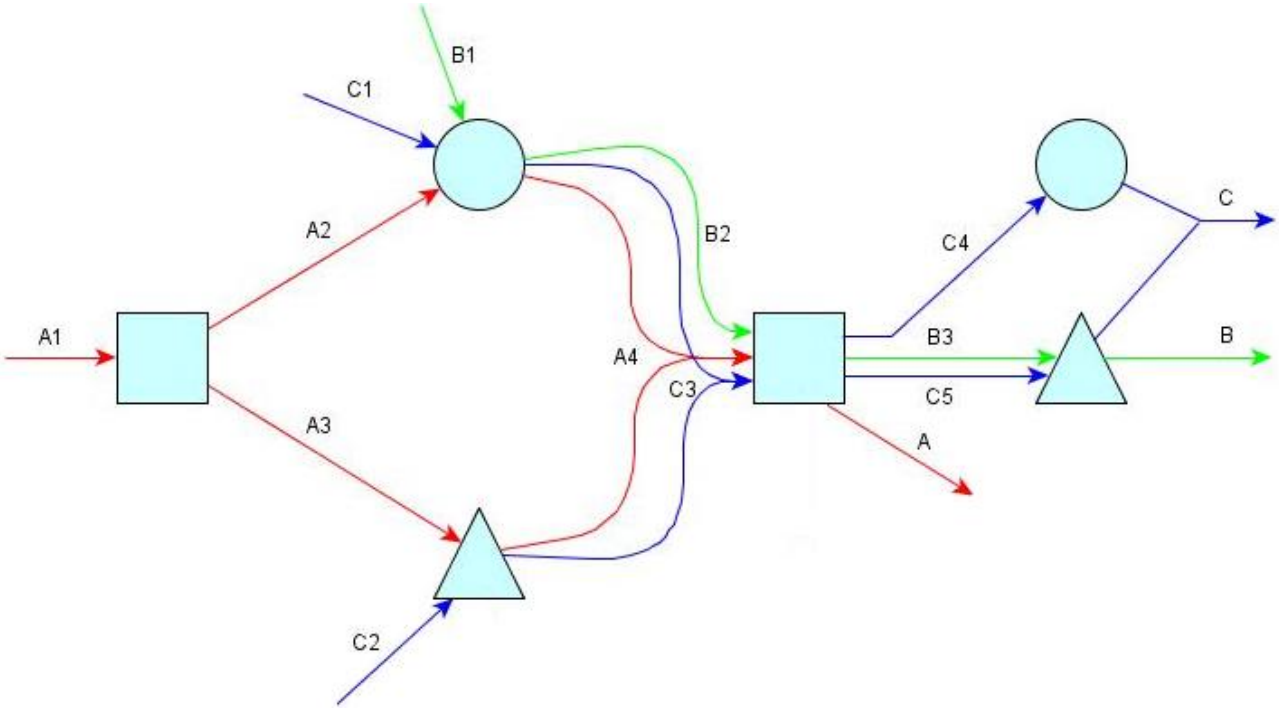


Рис. 4.12. Узагальнений граф передування завдань для множини виробів $\{A,B,C\}$

Іще один приклад побудови узагальненого технологічного графу наведено у додатку Д.

4.4.2.2 Алгоритм побудови узагальненої технологічної карти

Розглянемо принципову схему алгоритму побудови узагальненої технологічної карти.

Принципова схема алгоритму формування узагальненого графа

КРОК 1. Для кожного з відношень передування завдань, заданого графом G_i $(i=\overline{1,n})$ сформувати відповідний граф G_i^U, який має всі властивості узагальненого графа G. Ініціалізувати G пустим.

КРОК 2. ДЛЯ ВСІХ G_i^U, $i=\overline{1,n}$, ВИКОНУВАТИ

2.1. Створити чергу Q, у яку входитимуть завдання графа G_i^U, які необхідно послідовно додати до G. Ініціалізувати цю чергу усіма ребрами графа G_i^U, які входять у вершини, до яких не йде жодного ребра від інших вершин.

2.2. ПОКИ черга не пуста, ВИКОНУВАТИ

2.2.1. Отримати чергове ребро $e=\{v_1,v_2\}$ з Q, видаливши його з самої черги.

2.2.2. Додати дане ребро до графа G при умові збереження його властивості ациклічності, додаючи нові вершини, якщо це необхідно.

2.2.3. Видалити ребро e з графа G_i^U.

Якщо з v_1 у графі G_i^U більше не виходить жодне ребро – видалити v_1.

Якщо в v_2 у графі G_i^U більше не входить жодне ребро – видалити v_2.

2.2.4. Додати до черги Q всі ребра оновленого графа G_i^U, для яких виконується властивість, описана у п.2.1 (входження ребра у вершину, до якої не йде жодного ребра від іншої вершини).

Результатом роботи цього алгоритму G є узагальнений граф передування робіт вихідної множини виробів.

Далі розглянемо детальний алгоритм формування графа G.

Алгоритм формування узагальненого графа передування завдань

Вхідні дані. Множина ациклічних антитранзитивних графів $G_i=\langle V_i,E_i\rangle$ відношень безпосереднього передування завдань для всіх виробів $p_i\in P$, $i=\overline{1,n}$. У кожному графі G_i кожній вершині $v_{ij}\in V_i$, $j=\overline{1,g_i}$, ставиться у

відповідність виробнича дільниця $u(v_{ij}) \in U$ та номер завдання $j(v_{ij})$ з виготовлення виробу i.

Вихідні дані. Узагальнений граф проходження завдань $G = \langle V, E \rangle$.

Кожній вершині $v_q \in V, q = \overline{1,|V|}$ ставиться у відповідність виробнича дільниця $u(v_q) \in U$. Кожне ребро $e_{q_1 q_2} \in E$, $q_1, q_2 \in \{1,2,\ldots,|V|\}$, $q_1 \neq q_2$, асоціюється з індексом $i(e_{q_1 q_2}) \in \{1,2,\ldots,n\}$ виробу та індексом конкретного завдання $j(e_{q_1 q_2}) \in \{1,2,\ldots,g_{i(e_{q_1 q_2})}\}$ з виготовлення цього виробу.

Проміжні дані. Проміжні графи $G_i^U = \langle V_i^U, E_i^U \rangle$, $i = \overline{1,n}$, що є результатом приведення графів відношень передування завдань G_i до структури, відповідної узагальненому графу G.

Псевдокод алгоритму

```
Функція Узагальнений_граф(G_{i_1}, G_{i_2}, ..., G_{i_n})
1     G ← ∅
2     Для i = 1...n виконувати
3         G_i^U ← Проміжний_граф(G_i, i)
4         Об'єднати_графи(G, G_i^U)
5     Повернути G
```

Псевдокод допоміжних функцій, використаних у алгоритмі, наводиться далі.

Функція *Проміжний_граф* приймає на вхід граф, що має структуру відношення безпосереднього передування завдань, та індекс виробу, якому цей граф відповідає, і повертає граф зі структурою, яку мають проміжні графи та узагальнений граф G. Зауважимо, що завдання, які поступають у вершини такого графа «ззовні» (початкові завдання по виробах) також, без втрати загальності, вважаються ребрами графа. Для таких ребер в якості індексу вихідної вершини встановлюється значення `NIL`.

Функція *Проміжний_граф*($G_i = \langle V_i, E_i \rangle$, i)

1 $G_i^U = \langle V_i^U, E_i^U \rangle \leftarrow \varnothing$

2 Для $j = 1 \ldots g_i$ виконувати

3 $V_i^U \leftarrow V_i^U \cup$ Нова вершина v_{ij}^U : $u(v_{ij}^U) = u(v_{ij})$

4 *Вершина_початкова* ← TRUE

5 Для всіх ребер $(v_{ij*}, v_{ij}) \in E_i$

6 *Вершина_початкова* ← FALSE

7 $E_i^U \leftarrow E_i^U \cup$ Нове ребро $e_{j*,j}^U$: $i(e_{j*,j}^U) = i$, $j(e_{j*,j}^U) = j$

8 Якщо *Вершина_початкова* = TRUE

9 $E_i^U \leftarrow E_i^U \cup$ Нове ребро $e_{NIL,j}^U$: $i(e_{NIL,j}^U) = i$, $j(e_{NIL,j}^U) = j$

10 Повернути G_i^U

Процедура *Об'єднати_графи* приймає на вхід два графа, які мають структуру проміжного / узагальненого графа, та послідовно додає ребра другого графа до першого, додаючи нові вершини у перший граф, коли це необхідно, зберігаючи його властивість ациклічності та завжди об'єднуючи вершини, які вказують на одну і ту саму виробничу дільницю, коли це можливо. Дана процедура використовує чергу Q з ребер e другого графа. Для даних, що описують ребро e з цієї черги, використовуються наступні позначення:

- Src(e) – вершина, з якої ребро e виходить;
- Dest(e) – вершина, у яку ребро e входить;
- Prev(e) – множина ребер, що є попередниками e, тобто ребра, що входять у вершину Src(e) (якщо Src(e) ≠ NIL).

У множину Prev(e) також включаються всі ребра, видалені з другого графа в процесі виконання алгоритму. На практиці для збереження цих ребер зручно на етапі ініціалізації створити локальну копію графа, ребра з якого будуть видалятися, та отримувати множину Prev(e) для будь-якого e з неї.

Псевдокод процедури *Об'єднати_графи* наступний:

```
Процедура Об'єднати_графи(G = ⟨V,E⟩, G_i^U = ⟨V_i^U, E_i^U⟩)
1     Q ← ∅
2     Додати в Q всі ребра e* ∈ E_i^U, для яких Src(e*) = NIL
3     Поки Q ≠ ∅, виконувати
4         Отримати чергове ребро e з Q, видалити його з Q
5         Виконати_спеціальну_вставку_ребра(G, e, Prev(e))
6         Видалити ребро e з графа G_i^U
7         Якщо список вхідних ребер у вершину Dest(e) пустий
8             Додати в Q всі ребра e* ∈ E_i^U, для яких Src(e*) = Dest(e)
```

Допоміжна процедура *Виконати_спеціальну_вставку_ребра* призначена для додання ребра e в граф $G = \langle V, E \rangle$ за спеціальними правилами, що у деталях розкриваються нижче у відповідному фрагменті псевдокоду. Ці правила спрямовані на те, щоб у графі G з'явилося ребро, що йде від вершини, яка посилається на ту ж виробничу дільницю, що й вершина `Src(`e`)` у графі, з якого це ребро додається (якщо `Src(`e`) ≠ NIL`).

На вершину, з якої виходитиме додане ребро, накладається обмеження: вона повинна мати серед своїх вхідних завдань (пар «індекс виробу – індекс завдання», що представляються ребрами графа G) всі завдання, які відповідають ребрам з множини `Prev(`e`)`.

Додане ребро повинно входити у вершину, яка посилається на ту ж виробничу дільницю, що й вершина `Dest(`e`)` у графі, з якого додається ребро. Але, при цьому, обов'язково повинна зберігатися властивість ациклічності графа G: якщо в графі G не існує такої вершини $v_{Dest} \in V$, що $u(v_{Dest}) = u(\texttt{Dest}(e))$, або такі вершини існують, але всі вони передують знайденій вершині $v_{Src} \in V$, для якої $u(v_{Src}) = u(\texttt{Src}(e))$ і виконується згадане вище обмеження (за побудовою алгоритму, така вершина v_{Src} завжди буде знайдена), то в граф G в якості v_{Dest} додається нова вершина.

Також виклик даної процедури передбачає після додання ребра e у граф G перенаправлення всіх наявних ребер $e^* \in E$, для яких $i(e^*) = i(e)$ та $j(e^*) = j(e)$ (тобто ребер, асоційованих з тим же завданням, що й ребро e) у найбільш «пізню» вершину серед всіх `Dest(`e^*`)`. Це потрібно для коректного об'єднання ребер, що відповідають одному і тому самому завданню. Далі буде наведено приклад, який демонструє необхідність виконання таких дій.

У наведеному нижче псевдокоді процедури *Виконати_спеціальну_вставку_ребра* використовуються наступні змінні:

- v_{Src} – вершина графа G, з якої виходитиме ребро, що додається;
- v_{Dest} – вершина графа G, у яку входитиме ребро, що додається;
- e_G – ребро, що додається у граф G.

Для ребра e повинні бути визначені такі дані:

- $u($ `Src(`e`)` $)$ – виробнича дільниця, на яку посилається вершина, з якої виходить ребро e (якщо `Src(`e`) ≠ NIL`);
- $u($ `Dest(`e`)` $)$ – виробнича дільниця, на яку посилається вершина, у яку входить ребро e;
- `Prev(`e`)` – множина ребер, що входять у вершину, з якої виходить ребро e (якщо `Src(`e`) ≠ NIL`).

Таким чином, псевдокод процедури *Виконати_спеціальну_вставку_ребра* має наступний вигляд:

Процедура *Виконати_спеціальну_вставку_ребра*(G, e, `Prev(`e`)`)

1 Якщо `Src(`e`) = NIL`

2 $v_{Dest} \leftarrow$ *Знайти_вершину*(G, $u($ `Dest(`e`)` $)$)

3 Якщо v_{Dest} = NIL

4 $V \leftarrow V \cup$ Нова вершина v_{Dest} : $u(v_{Dest}) = u($ `Dest(`e`)` $)$

5 $E \leftarrow E \cup$ Нове ребро $e_{NIL,v_{Dest}}$: $i(e_{NIL,v_{Dest}}) = i(e)$, $j(e_{NIL,v_{Dest}}) = j(e)$

6 Якщо `Src(`e`) ≠ NIL`

7 $v_{Src} \leftarrow$ *Знайти_з_ребрами*(G, $u($ `Src(`e`)` $)$, `Prev(`e`)`)

```
8       v_Dest ← Знайти_не_передуючу( G , u( Dest( e ) ) , v_Src )
9       Якщо v_Dest = NIL
10          V ← V ∪ Нова вершина v_Dest : u(v_Dest) = u( Dest( e ) )
11      E ← E ∪ Нове ребро e_{v_Src,v_Dest} : i(e_{v_Src,v_Dest}) = i(e), j(e_{v_Src,v_Dest}) = j(e)
12      Синхронізувати_ребра_по_однакових_завданнях( G , e )
```

Функції *Знайти_вершину* та *Знайти_з_ребрами* застосовуються для пошуку у графі G найближчої вершини, що посилається на вказану виробничу дільницю u', причому друга функція є модифікацією першої: при пошуку вона накладає на знайдену вершину обмеження, що та повинна мати серед своїх вхідних ребер такі, що представляють всі завдання з заданої множини. У даній роботі для реалізації цих функцій застосовуються ідеї пошуку вшир. Використовується черга пошуку вшир Q_{BFS}, елементами якої є вершини графа G. Псевдокод пропонованої реалізації функції *Знайти_вершину* наступний:

```
Функція Знайти_вершину( G = ⟨V, E⟩ , u' )
1   Q_BFS ← ∅
2   Додати в Q_BFS всі v' ∈ V | ∀v'' ∈ V (v'', v') ∉ E
3   Поки Q_BFS ≠ ∅, виконувати
4       Отримати вершину v' з Q_BFS, видалити її з черги
5       Якщо u(v') = u'
6           Повернути v'
7       Додати в Q_BFS всі v'' ∈ V | (v', v'') ∈ E
8   Повернути NIL
```

Примітка. При доданні нових вершин у чергу Q_{BFS} для запобігання виконання зайвих операцій корисно виконувати перевірку, чи додавалась ця вершина у чергу раніше, і, якщо так, не додавати вершину в чергу. Оскільки наведений псевдокод спрямований на пояснення самої ідеї алгоритму, такі деталі реалізації у ньому опускаються.

Псевдокод функції *Знайти_з_ребрами* у реалізації, що пропонується в даній роботі, має такий вигляд:

Функція *Знайти_з_ребрами*($G=\langle V,E\rangle$, u' , $E*$)

1 $Q_{BFS} \leftarrow \varnothing$

2 Додати в Q_{BFS} всі $v' \in V \mid \forall v'' \in V \quad (v'',v') \notin E$

3 Поки $Q_{BFS} \neq \varnothing$, виконувати

4 Отримати вершину v' з Q_{BFS}, видалити її з черги

5 Якщо $u(v') = u'$

6 *Вершина_задовольняє_умові* ← TRUE

7 $E' \leftarrow \{e_G \in E \mid \text{Dest}(e_G) = v'\}$

8 Для всіх $e* \in E*$ виконувати

9 Якщо не існує $e_G \in E' : \; i(e_G) = i(e*) \wedge j(e_G) = j(e*)$

10 *Вершина_задовольняє_умові* ← FALSE

11 Вийти з циклу «Для всіх $e* \in E*$»

12 Якщо *Вершина_задовольняє_умові* = TRUE

13 Повернути v'

14 Додати в Q_{BFS} всі $v'' \in V \mid (v',v'') \in E$

15 Повернути NIL

Функція *Знайти_не_передуючу* (див. рядок 8 процедури *Виконати_спеціальну_вставку_ребра*) призначена для знаходження у графі $G=\langle V,E\rangle$ найближчої по відношенню до початкових вершин графа вершини, що посилається на виробничу дільницю u', але при цьому не є попередником вершини v', а також не співпадає з нею (це суттєво при доданні ребра, виробничі дільниці початку та кінця якого співпадають).

Псевдокод цієї функції можна представити наступним чином:

Функція *Знайти_не_передуючу*($G=\langle V,E\rangle$, u' , v')

1 Для кожної вершини $v* \in V$ виконувати

2 Якщо $u(v*) = u'$ і $v* \neq v'$

```
3            Якщо Вершина_передує(G, v*, v′) = FALSE
4                Повернути v*
5    Повернути NIL
```

Слід зазначити, що порядок обходу вершин графа G в рядку 1 наведеного псевдокоду є суттєвим: якщо існує декілька альтернативних вершин, що задовольняють умовам з рядків 2 і 3, то з функції потрібно повернути найбільш «ранню» (найближчу до початкових вершин графа) серед них. Тому для забезпечення обходу вершин у потрібному порядку з логічної точки зору було б прийнятним використання підходу, викладеного вище в описі функції *Знайти_вершину*, що заснований на використанні черги вершин Q_{BFS}. Але якщо вершини графа G мають свої унікальні індекси, причому індекси доданих пізніше вершин більші за індекси вершин, доданих раніше, то використання цього підходу у явному вигляді є надлишковим: легко побачити, що при такій індексації вершин для виконання умови повернення найбільш ранньої вершини серед тих, що задовольняють умовам рядків 2–3, достатнім є обхід вершин графа G в порядку зростання їх індексів.

Допоміжна функція *Вершина_передує* повертає значення TRUE, якщо у графі G існує шлях від вершини $v*$ до вершини v', або FALSE в протилежному випадку. Для реалізації цієї функції може бути використана велика кількість підходів. Розглянемо один з них, що був використаний у даній роботі. Він базується на ідеях пошуку вшир та використовує чергу Q_{BFS}, у яку додаються вершини, що проходяться. Проходження вершин здійснюється від v' у напрямку, протилежному напрямку ребер. Псевдокод такого варіанту реалізації даної функції представляється наступним чином:

```
Функція Вершина_передує(G = ⟨V, E⟩, v*, v′)
1    Q_BFS ← ∅
2    Додати в Q_BFS всі v″ ∈ V | (v″, v′) ∈ E
3    Поки Q_BFS ≠ ∅, виконувати
4        Отримати вершину v″ з Q_BFS, видалити її з черги
```

```
5           Якщо  v'' = v*
6               Повернути TRUE
7           Додати в  Q_BFS  всі  v''' ∈ V  |  (v''', v'') ∈ E
8       Повернути FALSE
```

Примітка. Як і у функціях *Знайти_вершину* та *Знайти_з_ребрами*, у даній функції доцільною є оптимізація, що передбачає перевірку при доданні деякої вершини у чергу, чи додавалася вона туди раніше – якщо так, додавати ще раз її не потрібно, оскільки це призведе до сповільнення виконання алгоритму.

Процедура *Синхронізувати_ребра_по_однакових_завданнях* (див. псевдокод процедури *Виконати_спеціальну_вставку_ребра*, рядок 12) необхідна для перенаправлення усіх ребер графа $G = \langle V, E \rangle$, що відповідають одному і тому ж самому завданню, в одну вершину.

Розглянемо приклад, який ілюструє необхідність такого перенаправлення ребер графа G у ситуаціях, коли одне завдання має декілька завдань-попередників. Нехай обслуговуюча система складається з двох виробничих дільниць, що на ілюстраціях позначаються квадратом та кругом відповідно, і повинна виготовити два вироби A і B, графи відношень передування завдань яких та відповідні їм проміжні графи G_A^U і G_B^U представлені на рисунку 4.13.

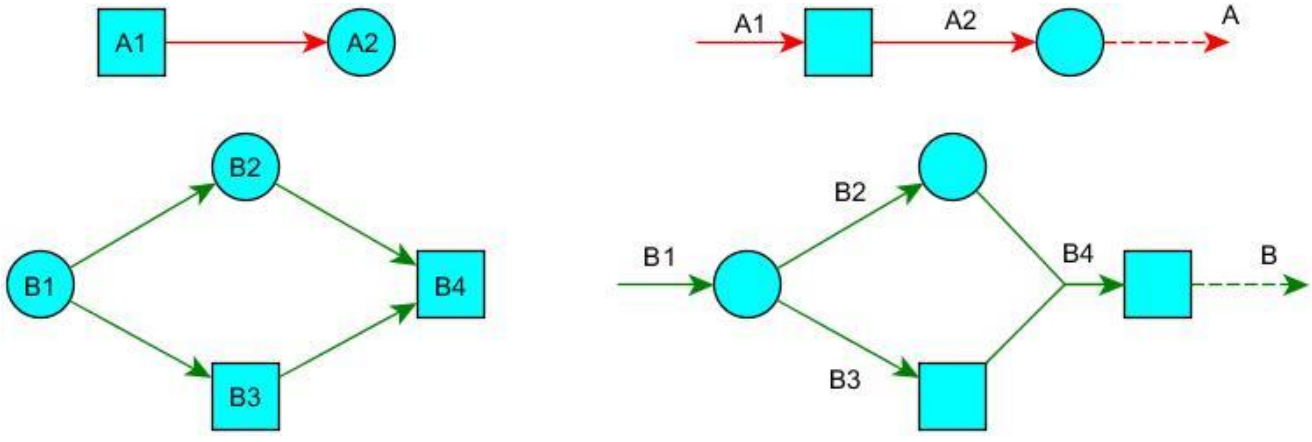


Рис. 4.13. Графи G_A, G_B (ліворуч) та G_A^U, G_B^U (праворуч)

Узагальнений граф G для цих двох виробів повинен мати вигляд, наведений на рисунку 4.14.

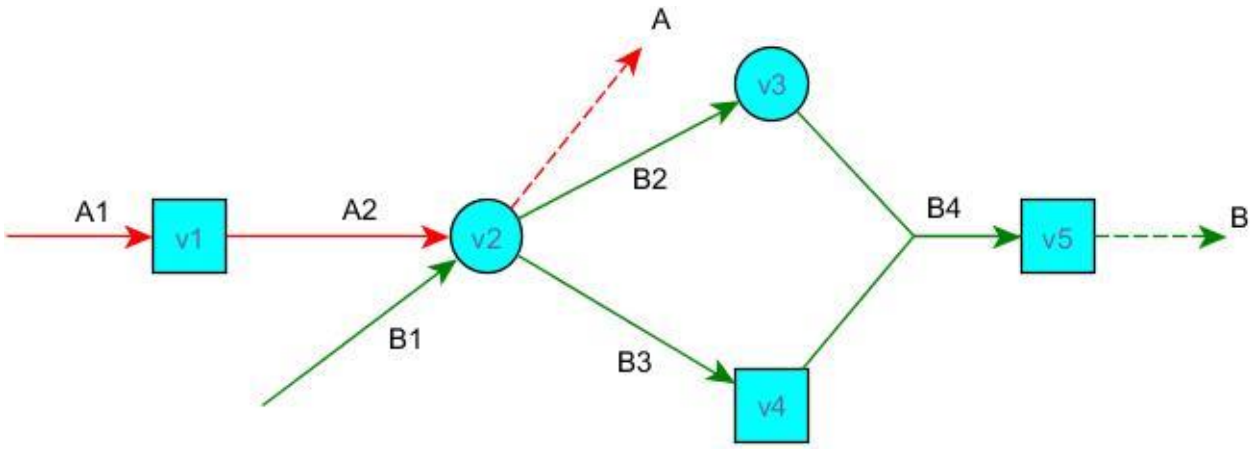


Рис. 4.14. Правильно побудований граф G для прикладу, що розглядається

Проте без застосування спеціальної процедури для узгодження ребер, які відповідають одному і тому самому завданню, при виконанні описаного алгоритму формування узагальненого графа виникла б помилка, ілюстрація якої наведена на рисунку 4.15.

Ребра, що відповідають завданню $B4$, входять у різні вершини і через це не зможуть бути визнані за одне ціле алгоритмом диспетчеризації складання розкладів, який працюватиме з даним узагальненим графом, обробляючи кожну його вершину разом з її вхідними ребрами відокремлено від інших вершин, що призведе до помилки: завдання $B4$ буде фігурувати у результуючому розкладі два рази.

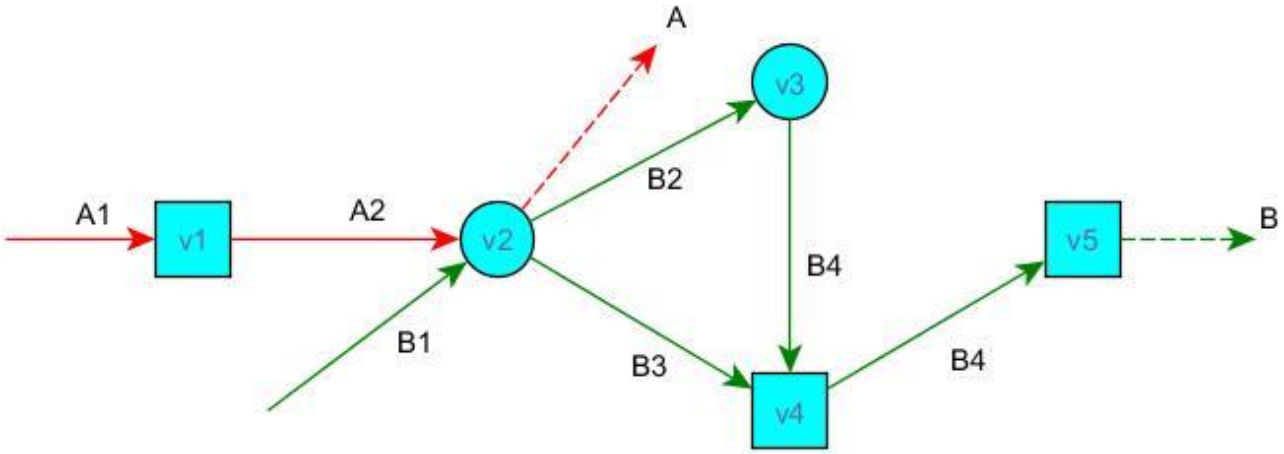


Рис. 4.15. Помилка в побудові узагальненого графа

Причина виникнення помилки очевидна: при доданні ребра, що відповідає завданню $B4$, яке виходить з вершини v_3 (тип виробничої дільниці «круг»),

функція *Знайти_не_передуючу* направила його у найближчу (по відношенню до початкової вершини v_1) вершину з типом виробничої дільниці «квадрат», від якої не існує шляху до v_3, тобто у вершину v_4. При доданні ребра $B4$ «квадрат»–«квадрат» в якості вихідної вершини була обрана v_4, а при визначенні вершини, в яке це ребро увійде, в граф була додана вершина v_5, до якої ребро й було направлене. Таким чином, ребра, що відповідають одному і тому самому завданню, виявилися направленими до різних вершин. Така ситуація є недопустимою.

Для запобігання виникненню помилок такого роду в алгоритм була введена процедура *Синхронізувати_ребра_по_однакових_завданнях*, яка перенаправляє усі ребра, що відповідають одному і тому самому завданню, в одну вершину. Очевидно, що для збереження властивості ациклічності графа цією вершиною повинна бути `Dest(`e'`)`, найбільш віддалена від початкових вершин графа, $\forall e' \in E \mid i(e') = i(e) \wedge j(e') = j(e)$.

В даній роботі використовується наступний варіант реалізації цієї процедури. Він базується на пошуку вшир починаючи з вершини `Dest(`e`)` в обидві сторони (у напрямку ребер графа та проти нього) всіх ребер e', для яких $i(e') = i(e) \wedge j(e') = j(e)$.

Визначимо для процедури такі допоміжні дані:

- $E*$ – множина ребер графа G, які відповідають тому самому завданню, що й ребро e;
- $v*$ – вершина, у яку будуть перенаправлені всі ребра з множини $E*$;
- Q_{BFS} – черга пошуку вшир, елементами якої є вершини графа G (див. примітку до функції *Знайти_вершину* щодо оптимізації завдання з даною чергою).

Псевдокод даної реалізації процедури синхронізації ребер у графі $G = \langle V, E \rangle$ по роботах, що відповідають ребру e, виглядає наступним чином:

Процедура *Синхронізувати_ребра_по_однакових_завданнях(*G *,* e*)*

```
1     E* ← ∅
2     v* ← Dest(e)
3     Додати в Q_BFS вершину Dest(e)
4     Поки Q_BFS ≠ ∅, виконувати
5         Отримати вершину v' з Q_BFS, видалити її з черги
6         Для всіх e' ∈ E | Dest(e') = v'
7             Якщо i(e') = i(e) ∧ j(e') = j(e)
8                 Якщо e' ∉ E*
9                     E* ← E* ∪ {e'}
10                Видалити ребро e' з графа G
11        Додати в Q_BFS всі v'' ∈ V | (v'', v') ∈ E
12    Додати в Q_BFS всі v' ∈ V | (Dest(e), v') ∈ E
13    Поки Q_BFS ≠ ∅, виконувати
14        Отримати вершину v' з Q_BFS, видалити її з черги
15        Для всіх e' ∈ E | Dest(e') = v'
16            Якщо i(e') = i(e) ∧ j(e') = j(e)
17                Якщо e' ∉ E*
18                    E* ← E* ∪ {e'}
19                Видалити ребро e' з графа G
20                v* ← v'
21        Додати в Q_BFS всі v'' ∈ V | (v', v'') ∈ E
22    Для всіх e' ∈ E* виконувати
23        E ← E ∪ Нове ребро e* = e' : Dest(e*) ← v*
```

Примітка. Можна побачити, що в прикладах побудови узагальненого графа, що наводилися раніше, у випадках, коли декілька паралельних вершин проміжного графа, які посилаються на одну й ту ж виробничу дільницю, під час побудови узагальненого графа об’єднуються в одну вершину, з цієї вершини виходить лише одне ребро (див., наприклад, рис. 4.5 і Д.1 дод.Д – вершини

з вхідними завданнями 1:4 та 1:5). Це відбувається завдяки тому, що розглянута вище реалізація процедури *Синхронізувати_ребра_по_однакових_ завданнях* (див. рядки 8 та 17 наведеного псевдокоду цієї процедури) «відфільтровує» ребра $e' \in E$, для яких $\exists\, e'' \in E^* \mid i(e') = i(e'') \ \wedge \ j(e') = j(e'') \wedge \mathtt{Src}(e') = \mathtt{Src}(e'') \wedge \mathtt{Dest}(e') = \mathtt{Dest}(e'')$. Розглянута реалізація даної процедури не додає ребра, що повторюються, у множину E^*, проте все одно видаляє їх з множини ребер E графа G. Слід зазначити, що така «фільтрація» на етапі побудови узагальненого графа не є обов'язковою, оскільки алгоритм диспетчеризації складання розкладу за узагальненим графом, про який мова йтиме далі, враховує лише ребра, які відповідають різним завданням.

Загальна примітка. Графи G_i^U, $i = \overline{1,n}$, є лише проміжними структурами даних. Припущення про те, що для випадку $n = 1$ граф G_1^U співпадає з узагальненим графом G, є помилковим. Ілюстрація цього наведена на рисунку 4.16.

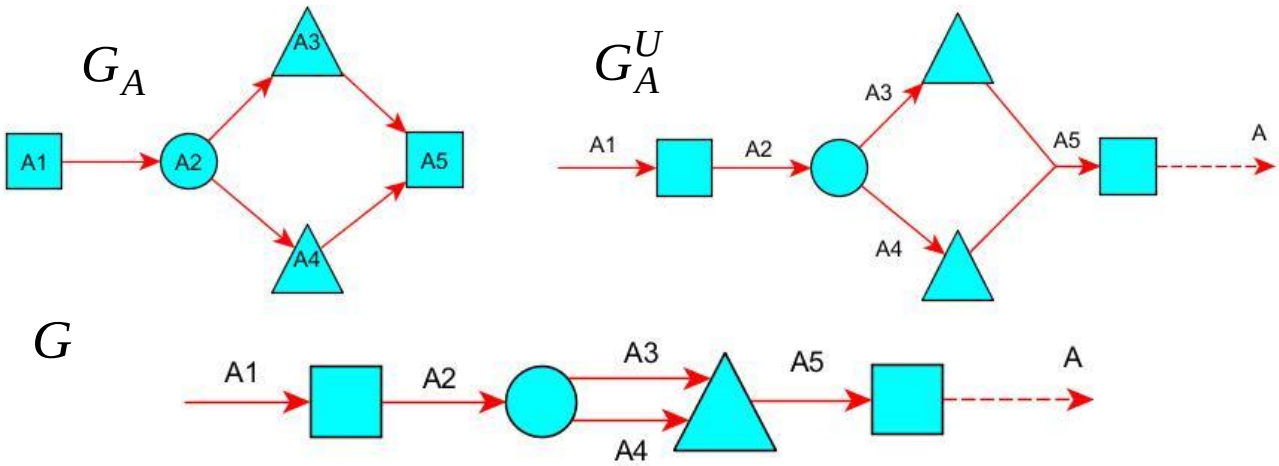


Рис. 4.16. Приклад, що показує відмінність графа G_A (G_A^U) від узагальненого графа G_A при кількості виробів G_A

4.4.3 Задача диспетчеризації складання розкладів виконання завдань на окремих дільницях за узагальненою технологічною картою

Складання розкладу здійснюється на основі узагальненого графа. Процес починається з його кінця (з строкових вузлів графа) і просувається в напрямку, протилежному до напрямку ребер, згідно з алгоритмом, що розкривається далі.

Алгоритм складання загального розкладу виконання завдань за узагальненою технологічною картою працює згідно з наступною принциповою схемою.

Принципова схема алгоритму

КРОК 1. Визначити множину строкових вузлів.

КРОК 2. ПОКИ множина строкових вузлів не пуста ВИКОНУВАТИ:

2.1. Отримати черговий вузол з множини строкових вузлів.

2.2. Визначити множину завдань, які повинні бути обслужені поточним вузлом.

2.3. Скласти розклад виконання завдань поточним вузлом.

2.4. Визначити директивні строки виконання завдань.

Вибір алгоритму, що буде використаний при складанні розкладу виконання завдань вузла i, залежить від:

- кількості пристроїв вузла;
- ідентичності/неідентичності пристроїв вузла;
- того, чи є директивний строк спільним для всіх завданьт вузла;
- наявності/відсутності вимоги щодо компактності результуючого розкладу (компактним будемо називати розклад виконання завдань на дільниці, у якому відсутні перерви у роботі обладнання, тобто при переході до наступного завдання пристрої не простоюють).

Розглянемо алгоритм складання розкладу за узагальненою технологічною картою більш формально.

Алгоритм складання розкладу за узагальненою технологічною картою

Вхідні дані. Вхідними даними алгоритму є:

– узагальнений граф проходження завдань $G = \langle V, E \rangle$ (кожній вершині $v_q \in V, q = \overline{1,|V|}$ ставиться у відповідність виробнича дільниця $u(v_q) \in U$, кожне ребро $e_{q_1 q_2} \in E$, $q_1, q_2 \in \{1,2,\ldots,|V|\}$, $q_1 \neq q_2$, асоціюється з індексом $i(e_{q_1 q_2}) \in \{1,2,\ldots,n\}$ виробу та індексом конкретного завдання $j(e_{q_1 q_2}) \in \{1,2,\ldots,g_{i(e_{q_1 q_2})}\}$ з виготовлення цього виробу);

– множина $M^U = \{M_y\}$ векторів коефіцієнтів продуктивності пристроїв $M_y = \{k_{yl}\}$ ($l = \overline{1,m_y}$, де m_y – кількість пристроїв на дільниці u_y) для всіх виробничих дільниць $u_y \in U$, $y = \overline{1,u}$, де $u = |U|$;

– множина директивних строків виготовлення виробів $D = \{d_i\}$, $i = \overline{1,n}$;

– множина «еталонних» тривалостей виконання завдань $T = \{t_{ij}\}$, $i = \overline{1,n}$, $j = \overline{1,g_i}$.

Вихідні дані. Вихідними даними алгоритму є розклад $S = \{s_{ij}\}$, $i = \overline{1,n}$, $j = \overline{1,g_i}$, що представляється у вигляді множини окремих записів по кожного із завдань. Кожний такий запис s_{ij} включає:

– `prd(`s_{ij}`)` – індекс виробу $i \in \{1,2,\ldots,n\}$, роботі якого присвячений даний запис;

– `job(`s_{ij}`)` – індекс цього завдання $j \in \{1,2,\ldots,g_{\texttt{prd}(s_{ij})}\}$ в межах технологічної карти виробу `prd(`s_{ij}`)`;

– `sec(`s_{ij}`)` – посилання на виробничу дільницю $u_{ij} \in U$, на якій виконується завдання;

– `mchn(`s_{ij}`)` – індекс пристрою виробничої дільниці `sec(`s_{ij}`)`, якій дане завдання було призначене;

– `bgn(`s_{ij}`)` – момент початку виконання завдань;

– `end(`s_{ij}`)` – момент завершення завдань.

У псевдокоді, що буде наводитися далі, позначатимемо додання нового запису до розкладу як $S \leftarrow S \cup s($`prd(s), job(s), sec(s), mchn(s), bgn(s), end(s)`$)$.

Проміжні дані. Проміжними даними алгоритму є:

– множина директивних строків виконання завдань $D^J = \{d_{ij}^J\}$, $i = \overline{1,n}$, $j = \overline{1,g_i}$;

– черга Q з вершин (вузлів) $v \in V$ узагальненого графа $G = \langle V, E \rangle$, для вхідних завдань яких вже були визначені директивні строки (а відтак, для яких можна складати розклад);

– g^v – кількість різних завдань, що входять у черговий вузол узагальненого графа;

– $I^v = \{i_w^v\}$, $w = \overline{1,g^v}$ – вектор індексів виробів, завданнями з виготовлення яких є завдання, що входять у черговий вузол ($i_w^v \in \{1,2,\ldots,n\}$);

– $J^v = \{j_w^v\}$, $w = \overline{1,g^v}$ – вектор індексів завдань, що входять у черговий вузол, в межах технологічних карт своїх виробів ($j_w^v \in \{1,2,\ldots,g_{i_w^v}\}$);

– $T^v = \{t_w^v\}$, $w = \overline{1,g^v}$ – вектор «еталонних» тривалостей завдань, що входять у черговий вузол;

– $D^v = \{d_w^v\}$, $w = \overline{1,g^v}$ – вектор директивних строків завдань, що входять у черговий вузол;

– $L^v = \{\lambda_w^v\}$, $w = \overline{1,g^v}$ – вектор списків індексів вершин, з яких надходять завдання, що входять у черговий вузол v: кожний елемент $\lambda_w^v \in L^v$ являє собою список індексів вершин графа G, з яких завдання j_w^v з виготовлення виробу i_w^v надійшло у вершину v;

– $B^v = \{b_w^v\}$, $w = \overline{1,g^v}$ – вектор моментів початку завдань, що входять у черговий вузол;

– $M^v = \{l_w^v\}$, $w = \overline{1, g^v}$ – вектор індексів пристроїв виробничої дільниці $u(v)$, на яких у розкладі виконуватимуться завдання, що входять у черговий вузол v: $l_w^v \in \{1,2,\ldots,m(u(v))\}$, де $m(u(v))$ – кількість пристроїв на дільниці $u(v)$.

Примітка. При доданні елемента в чергу Q необхідно перевіряти, чи додавався він у цю чергу раніше – якщо так, повторне додання вершини в чергу здійснюватися не повинно. Під «отриманням вершини з черги» розуміється її отримання з наступним видаленням з самої черги.

У псевдокоді, що наводиться нижче, справедливі всі позначення, введені у п. 4.3.1.

Псевдокод алгоритму

```
Функція Скласти_розклад(G = ⟨V,E⟩, M^U, D, T)
1     S ← ∅
2     Для всіх d_ij^J ∈ D^J виконувати
3         d_ij^J ← ∞
4     Ініціалізувати_дир_терм_останніх_завдань(G, D, D^J)
5     Q ← Строкові_вершини(G)
6     Поки Q ≠ ∅, виконувати
7         Отримати чергову вершину v ∈ V з Q
8         Сформувати_проміжні_дані(v, G, D^J, T, g^v, I^v, J^v, T^v, D^v, L^v, B^v, M^v)
9         Скласти_розклад_для_вузла(v, G, M^U, S, g^v, I^v, J^v, T^v, D^v, L^v, B^v, M^v)
10        Передати_нові_дир_строки(v, G, D^J, g^v, I^v, L^v, B^v)
11        Видалити з E всі e*:Dest(e*) = v
12        Q ← Q ∪ Строкові_вершини(G)
13    Повернути S
```

Процедура *Ініціалізувати_дир_строки_останніх_завдань* закріплює за директивними строками останніх завдань з виготовлення своїх виробів значення директивних строків цих виробів. Директивні строки решти

завдань, які не є останніми для своїх виробів, на цьому етапі залишаються невизначеними і мають початкове нескінченно велике значення.

Процедура *Ініціалізувати_дир_стр_останніх_завдань*(G, D, D^J)

1 Для всіх $v' \in V$ виконувати

2 Для всіх $e' \in E \mid \mathrm{Dest}(e') = v'$ виконувати

3 Якщо $\forall e'' \in E \mid \mathrm{Src}(e'') = v' \quad i(e'') \neq i(e')$

4 $d^J_{i(e'),\, j(e')} \leftarrow d_{i(e')}$

Функція *Строкові_вершини* повертає всі вершини графа G, з яких не виходить жодного ребра і які при цьому мають хоча б одне вхідне ребро.

Функція *Строкові_вершини*$(G = \langle V, E \rangle)$

1 $Q \leftarrow \varnothing$

2 Для всіх $v' \in V$ виконувати

3 Якщо $\forall e' \in E \quad \mathrm{Src}(e') \neq v'$

4 Якщо $\exists e'' \in E : \mathrm{Dest}(e'') = v'$

5 $Q \leftarrow Q \cup \{v'\}$

6 Повернути Q

Процедура *Сформувати_проміжні_дані* визначає кількість g^v різних завдань, що входять у вузол v (вхідні ребра, які відповідають одному і тому самому завданню, об'єднуються та інтерпретуються як одне вхідне завдання), та заповнює вектори I^v, J^v, T^v, D^v, L^v належними значеннями. Вектори B^v та M^v є вихідними даними для процедур розподілення завдань за пристроями на окремих вузлах узагальненого графа, проте вони ініціалізуються у цій процедурі.

Процедура *Сформувати_проміжні_дані*$(v, G, D^J, T, g^v, I^v, J^v, T^v, D^v, L^v, B^v, M^v)$

1 $g^v \leftarrow 0, \quad I^v, J^v, T^v, D^v, L^v, B^v, M^v \leftarrow \varnothing$

2 Для всіх $e \in E : \mathrm{Dest}(e) = v$ виконувати

3 Якщо $\exists w^* : i^v_{w^*} = i(e) \wedge j^v_{w^*} = j(e)$

4 $\lambda^v_{w^*} \leftarrow \lambda^v_{w^*} \cup \{\mathrm{Src}(e)\}$

5 Інакше

6 $g^v \leftarrow g^v + 1$

7 $i^v_{g^v} \leftarrow i(e)$

8 $j^v_{g^v} \leftarrow j(e)$

9 $t^v_{g^v} \leftarrow t_{i(e),j(e)}$

10 $d^v_{g^v} \leftarrow d^J_{i(e),j(e)}$

11 $\lambda^v_{g^v} \leftarrow$ {Src(e)}

12 Для $w=1\ldots g^v$ виконувати

13 $b^v_w \leftarrow \infty$

14 $l^v_w \leftarrow$ NIL

Примітка. Операція інкременту значення g^v (див. рядок 6 псевдокоду процедури *Сформувати_проміжні_дані*) має бути перенесена в кінець ітерації у випадках, коли нумерація починається з нуля, а не з одиниці. Також в даному псевдокоді опускаються моменти, які стосуються управління пам'яттю для векторів, над якими виконується робота.

Процедура *Скласти_розклад_для_вузла* здійснює виклик однієї з процедур, які заповнюють вектори B^v (моменти початку виконання вхідних завдань) та M^v (індекси пристроїв виробничої дільниці $u(v)$, на яких виконуватимуться вхідні завдання), таким чином складаючи розклад для вузла v узагальненого графа G. Відповідні записи вносяться у загальний розклад S. Конкретна процедура, що розподілятиме завдання за пристроями дільниці та визначатиме моменти початку їх виконання, залежить від кількості пристроїв $m(u(v))$ на виробничій дільниці $u(v)$, виконання або невиконання умови рівності всіх елементів вектора $M(u(v))$ коефіцієнтів продуктивності цих пристроїв,

виконання або невиконання умови рівності всіх директивних строків $d_w^v \in D^v$ ($w = \overline{1, g^v}$) завдань, що входять у вузол, тощо.

Така процедура повинна приймати на вхід наступні параметри:

– $m(u(v))$ – кількість пристроїв на виробничій дільниці;

– $M(u(v))$ – вектор коефіцієнтів продуктивності пристроїв $1,2,\ldots,m(u(v))$;

– g^v – кількість завдань, що мають бути розподілені за пристроями виробничої дільниці;

– T^v – вектор тривалостей виконання завдань $1,2,\ldots,g^v$ на «еталонному» пристрої;

– D^v – вектор директивних строків виконання завдань $1,2,\ldots,g^v$;

– B^v – вектор моментів початку виконання завдань $1,2,\ldots,g^v$;

– M^v – вектор індексів пристроїв виробничої дільниці $u(v)$, на яких завдання $1,2,\ldots,g^v$ будуть виконуватися.

Останні два своїх аргументи процедура розподілення завдань повинна заповнити – на момент її виклику вектор B^v заповнено нескінченно великими числами, а вектор M^v – значеннями `NIL`.

Схематично процедуру *`Скласти_розклад_для_вузла`* можна представити у наступному вигляді:

`Процедура` *`Скласти_розклад_для_вузла`*$(v, G, M^U, S, g^v, I^v, J^v, T^v, D^v, L^v, B^v, M^v)$

1 *`Тип_вузла`* $\leftarrow$ *`Визначити_тип_вузла`*$(v, G, M^U, g^v, I^v, J^v, T^v, D^v, L^v, \ldots)$

2 *`Обрана_процедура`* $\leftarrow$ *`Проц_розпод_завдань`*(*`Тип_вузла`*)

3 `Виконати` *`Обрана_процедура`*$(m(u(v)), M(u(v)), g^v, T^v, D^v, B^v, M^v)$

4 `Для` $w = 1 \ldots g^v$ `виконувати`

5 $S \leftarrow S \cup s(i_w^v, j_w^v, u(v), l_w^v, b_w^v, b_w^v + \frac{t_w^v}{k(u(v))_{l_w^v}})$

Структура змінної *Тип_вузла*, а також псевдокод функцій *Визначити_тип_вузла* та *Проц_розпод_завдань* не розкриваються. Ці дані залежать від наявних алгоритмів розподілення завдань за пристроями та їх вимог щодо особливостей вузла, для якого вони застосовуються. Вимоги до процедури, що повертається з функції *Проц_розпод_завдань*, були наведені вище.

Процедура *Передати_нові_дир_строки* призначена для передачі визначених після виконання процедури *Скласти_розклад_для_вузла* моментів початку виконання завдань (вектор B^v) завданням-попередникам як їх нових потенційних директивних строків. Фактичний директивний строк виконання завдання визначається як мінімальне зі значень, які їй передавалися.

Процедура *Передати_нові_дир_строки*($v, G, D^J, g^v, I^v, L^v, B^v$)

1 Для $w = 1 \ldots g^v$ виконувати

2 Для всіх $v' \in \lambda_w^v \mid v' \neq$ NIL виконувати

3 Для всіх $e' \in E \mid \text{Dest}(e') = v'$ виконувати

4 Якщо $i(e') = i_w^v$

5 $d^J_{i(e'),j(e')} \leftarrow \min(d^J_{i(e'),j(e')}, b_w^v)$

4.5 Огляд програмної реалізації інструментального комплексу ІТОКПДВ

Для планування роботи підприємства та застосування попередньо наведених задач, у рамках розробки інструментального комплексу інформаційної технології, були реалізовані наступні функції:

- ведення дільниць;
- ведення обладнання;
- ведення завдань та їх тривалостей виконання;
- ведення технологічних карт;
- реєстрація та обробка замовлень на виготовлення виробів;
- ведення об’ємного плану виробництва;

– складання плану завантаження потужностей:
  1) формування узагальненого технологічного графу проходження завдань по виробничих дільницях;
  2) диспетчеризація складання розкладів виконання завдань на окремих дільницях за узагальненим технологічним графом;
  3) складання розкладу виконання завдань для виробничої дільниці;
  4) складання розкладу виконання завдань для окремого пристрою виробничої дільниці;
– формування звітності:
  1) формування виробничого плану об’ємного календарного планування виробництва з врахуванням поточних замовлень;
  2) формування календарного плану виробництва у розрізі виробів (для окремого виробу);
  3) формування календарного плану роботи обладнання (для окремої дільниці);
  4) графіки відображення фактичного виконання плану, тощо.

Розглянемо процес роботи програмного продукту від ролі адміністратора. Перейшовши на головну сторінку застосування (рис. 4.17) адміністратор має доступ до всього функціоналу програмного продукту та йому відображається поточна завантаженість дільниць та поточні замовлення (тобто ті, які знаходяться у процесі виконання).

Поточна завантаженість дільниць розміщена для кожної дільниці в окремій вкладці, де відображається у графічному вигляді завантаженість пристроїв дільниці завданнями для виготовлення виробів (завдання відмінні за кольорами відповідно до замовлення). Для кожного пристрою відображається час завершення його роботи. При надходженні нових завдань на виконання виконується перепланування завантаженості пристроїв.

Поточні замовлення також розміщені в окремих вкладках (при надходженні нових замовлень або завершенні виконання існуючих вкладки

будуть додаватися або оновлюватися), де відображається у графічному вигляді стан перебігу виконання замовлення на виробництві (на кожній із дільниць).

Рис. 4.17. Головне вікно програмного продукту

Для повноцінного функціонування програмного продукту адміністратор заносить базову інформацію на сторінці «Введення даних» (рис. 4.18). Адміністратор вводить інформацію про вироби, які доступні для замовлення, інформацію для формування технологічних карт, інформацію про виробничі дільниці, пристрої, операції.

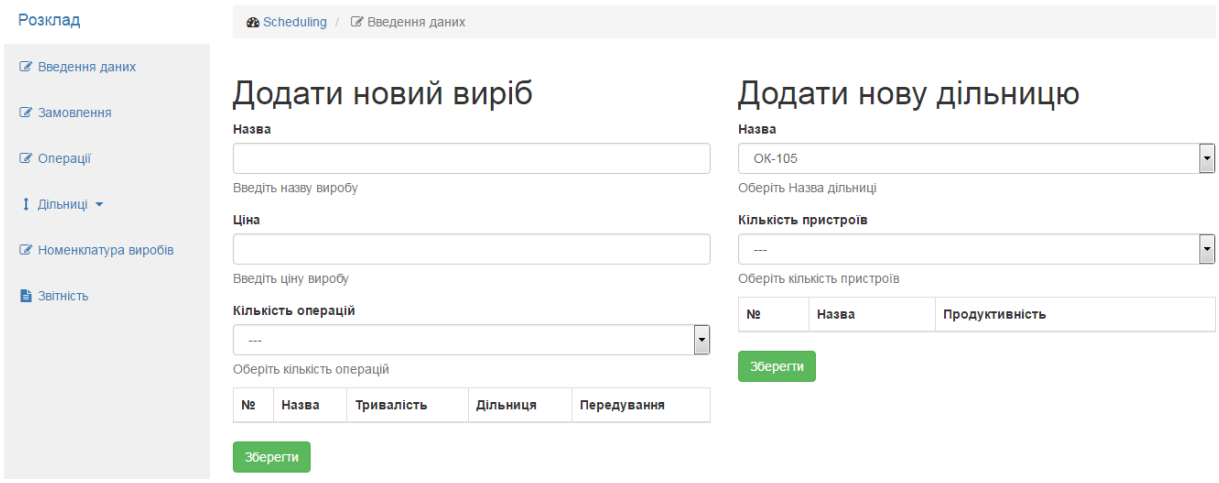


Рис. 4.18. Сторінка «Введення даних»

Створити нове замовлення, переглянути чи відредагувати існуючі можливо в пункті меню «Замовлення» (рис. 4.19).

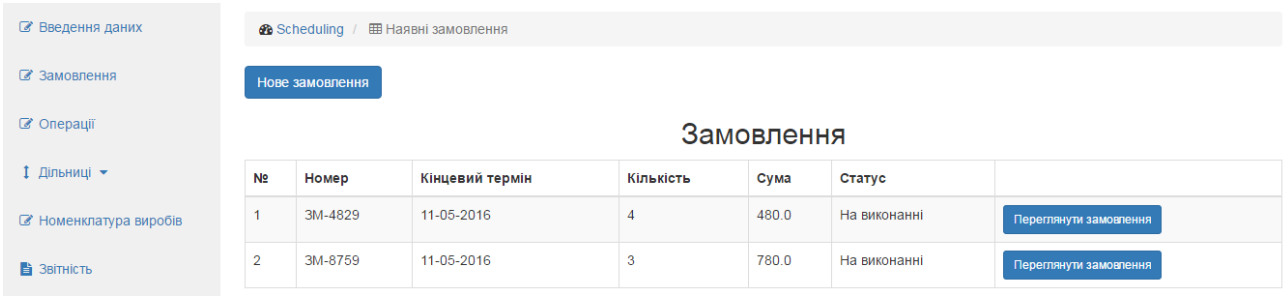


Рис. 4.19. Сторінка «Замовлення»

Переглянути інформацію про операції, з яких формуються технологічні карти можливо в пункті меню «Операції» (рис. 4.20).

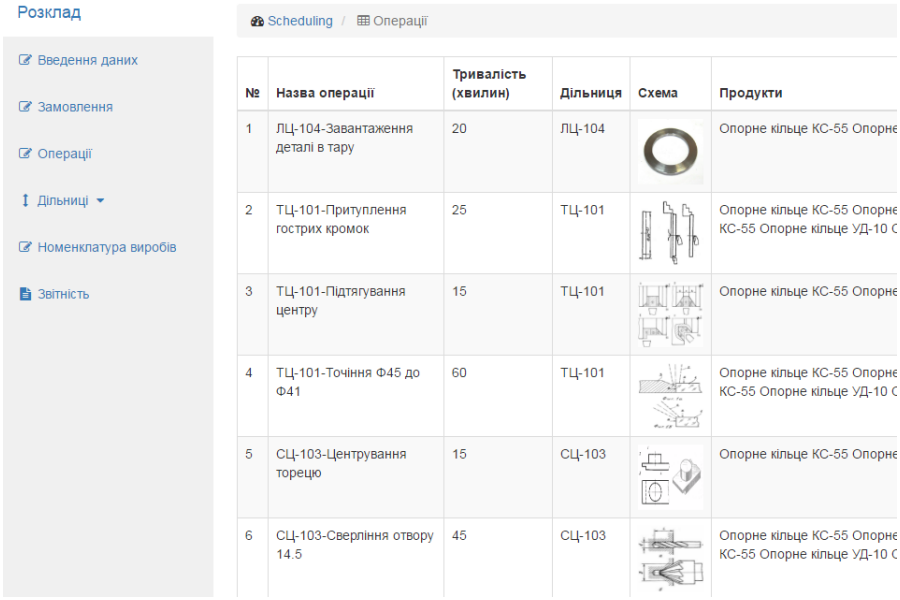


Рис. 4.20. Сторінка «Операції»

Переглянути інформацію про номенклатуру виробів, де відображається інформація по технологічних картах можна в пункті меню «Номенклатура виробів».

Інформація про дільниці та пристрої розміщена в пункті меню «Дільниці», де можна переглянути розклад роботи всіх дільниць та детальну інформацію про завантаженість пристроїв кожної дільниці.

Після успішного складання розкладу та виконання операцій адміністратор має можливість переглянути звітну інформацію в табличному та графічному

вигляді. Для цього в меню «Формування звітності» йому необхідно обрати відповідний звіт або графік (рис. 4.21).

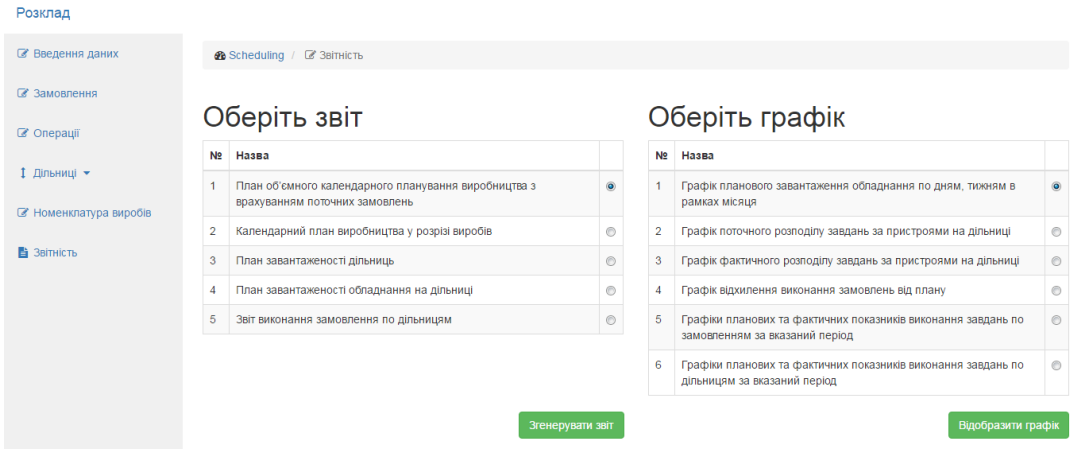


Рис. 4.21. Сторінка «Формування звітності»

Звіти подають у форматі «.xlsx», після чого їх можна відкрити за допомогою Microsoft Excel представимо деякі на рис. 4.22 - 4.24.

План виробництва на травень 2016 року					
№ з/п	***Номер замовлення***	***Код виробу***	***Ціна за одиницю, грн.***	***Кількість***	***Всього, грн.***
1	ЗМ-8759	ВР--3463463462	247,05	24	5 929,20
2	ЗМ-8760	ВР--1423533132	424,00	15	6 360,00
3	ЗМ-8761	ВР--3463463462	247,05	25	6 176,25
4	ЗМ-8762	ВР--1423533132	424,00	125	53 000,00
5	ЗМ-8763	ВР--3463463462	247,05	23	5 682,15
6	ЗМ-8764	ВР--1423533132	424,00	25	10 600,00
7	ЗМ-8765	ВР--3463463462	247,05	50	12 352,50
8	ЗМ-8766	ВР--1423533132	424,00	150	63 600,00
9	ЗМ-8767	ВР--3463463455	254,65	30	7 639,50
10	ЗМ-8768	ВР--3463463455	254,65	15	3 819,75
11	ЗМ-8769	ВР--3463463462	247,05	80	19 764,00
12	ЗМ-8770	ВР--1423533132	424,00	100	42 400,00
			Всього	***662***	***237323,35***

Рис. 4.22. План виробництва за вказаний період

№ з/п	***Номер замовлення***	***Код виробу***	***Назва виробу***	***Кількість***	Завершення виконання замовлення	Видача замовлення замовнику
1	ЗМ-4829	ВР--1179854084	Опорне кільце УД-10	4	11.05.2016 16:00	11.05.2016 17:00
2	ЗМ-8759	ВР--1179854084	Опорне кільце УД-10	2	11.05.2016 18:00	11.05.2016 8:00

Рис. 4.23. Календарний план виробництва

№ з/п	Номер замовлення	Код виробу	Назва виробу	Кількість	СЦ-103		ЛЦ-104		ТЦ-101		ПЦ-102		Завершення виконання замовлення	Видача замовлення замовнику
					Початок роботи	Завершення роботи	Початок роботи	Завершення роботи	Початок роботи	Завершення роботи	Початок роботи	Завершення роботи		
1	ЗМ-8759	ВР--1423533132	Опорне кільце RC-55	5	07.05.2016	08.05.2016	10.05.2016	11.05.2016	08.05.2016	09.05.2016	07.05.2016	08.05.2016	11.05.2016	11.05.2016
	ЗМ-8759	ВР--3463463462	Опорне кільце УД-10	4	08.05.2016	09.05.2016	11.05.2016	12.05.2016	09.05.2016	10.05.2016	08.05.2016	09.05.2016	12.05.2016	12.05.2016
2	ЗМ-4829	ВР--1423533132	Опорне кільце RC-55	5	09.05.2016	10.05.2016	12.05.2016	13.05.2016	10.05.2016	11.05.2016	09.05.2016	10.05.2016	13.05.2016	13.05.2016
	ЗМ-4829	ВР--3463463462	Опорне кільце УД-10	2	10.05.2016	11.05.2016	13.05.2016	14.05.2016	11.05.2016	12.05.2016	10.05.2016	11.05.2016	14.05.2016	14.05.2016
3	ЗМ-7088	ВР--1423533132	Опорне кільце RC-55	7	11.05.2016	12.05.2016	14.05.2016	15.05.2016	12.05.2016	13.05.2016	11.05.2016	12.05.2016	15.05.2016	15.05.2016
	ЗМ-7088	ВР--3463463462	Опорне кільце УД-10	2	12.05.2016	13.05.2016	15.05.2016	16.05.2016	13.05.2016	14.05.2016	12.05.2016	13.05.2016	16.05.2016	16.05.2016
4	ЗМ-7950	ВР--1423533132	Опорне кільце RC-55	3	13.05.2016	14.05.2016	16.05.2016	17.05.2016	14.05.2016	15.05.2016	13.05.2016	14.05.2016	17.05.2016	17.05.2016
	ЗМ-7950	ВР--3463463462	Опорне кільце УД-10	7	14.05.2016	15.05.2016	17.05.2016	18.05.2016	15.05.2016	16.05.2016	14.05.2016	15.05.2016	18.05.2016	18.05.2016
5	ЗМ-8812	ВР--1423533132	Опорне кільце RC-55	4	15.05.2016	16.05.2016	18.05.2016	19.05.2016	16.05.2016	17.05.2016	15.05.2016	16.05.2016	19.05.2016	19.05.2016
	ЗМ-8812	ВР--3463463462	Опорне кільце УД-10	5	16.05.2016	17.05.2016	19.05.2016	20.05.2016	17.05.2016	18.05.2016	16.05.2016	17.05.2016	20.05.2016	20.05.2016
6	ЗМ-8812	ВР--1423533132	Опорне кільце RC-55	1	17.05.2016	18.05.2016	20.05.2016	21.05.2016	18.05.2016	19.05.2016	17.05.2016	18.05.2016	21.05.2016	21.05.2016
	ЗМ-8812	ВР--3463463462	Опорне кільце УД-10	3	18.05.2016	19.05.2016	21.05.2016	22.05.2016	19.05.2016	20.05.2016	18.05.2016	19.05.2016	22.05.2016	22.05.2016
7	ЗМ-8812	ВР--1423533132	Опорне кільце RC-55	6	19.05.2016	20.05.2016	22.05.2016	23.05.2016	20.05.2016	21.05.2016	19.05.2016	20.05.2016	23.05.2016	23.05.2016
	ЗМ-8812	ВР--3463463462	Опорне кільце УД-10	8	20.05.2016	21.05.2016	23.05.2016	24.05.2016	21.05.2016	22.05.2016	20.05.2016	21.05.2016	24.05.2016	24.05.2016
8	ЗМ-8812	ВР--1423533132	Опорне кільце RC-55	2	21.05.2016	22.05.2016	24.05.2016	25.05.2016	22.05.2016	23.05.2016	21.05.2016	22.05.2016	25.05.2016	25.05.2016
	ЗМ-8812	ВР--3463463462	Опорне кільце УД-10	6	22.05.2016	23.05.2016	25.05.2016	26.05.2016	23.05.2016	24.05.2016	22.05.2016	23.05.2016	26.05.2016	26.05.2016
9	ЗМ-8812	ВР--1423533132	Опорне кільце RC-55	4	23.05.2016	24.05.2016	26.05.2016	27.05.2016	24.05.2016	25.05.2016	23.05.2016	24.05.2016	27.05.2016	27.05.2016
	ЗМ-8812	ВР--3463463462	Опорне кільце УД-10	3	24.05.2016	25.05.2016	27.05.2016	28.05.2016	25.05.2016	26.05.2016	24.05.2016	25.05.2016	28.05.2016	28.05.2016
10	ЗМ-8812	ВР--1423533132	Опорне кільце RC-55	9	25.05.2016	26.05.2016	28.05.2016	29.05.2016	26.05.2016	27.05.2016	25.05.2016	26.05.2016	29.05.2016	29.05.2016
	ЗМ-8812	ВР--3463463462	Опорне кільце УД-10	5	26.05.2016	03:34 07.05	29.05.2016	30.05.2016	27.05.2016	28.05.2016	26.05.2016	27.05.2016	30.05.2016	30.05.2016

Рис. 4.24. Календарний план виробництва у розрізі виробів

Згенеровані графіки відображаються в новій вкладці (рис. 4.25).

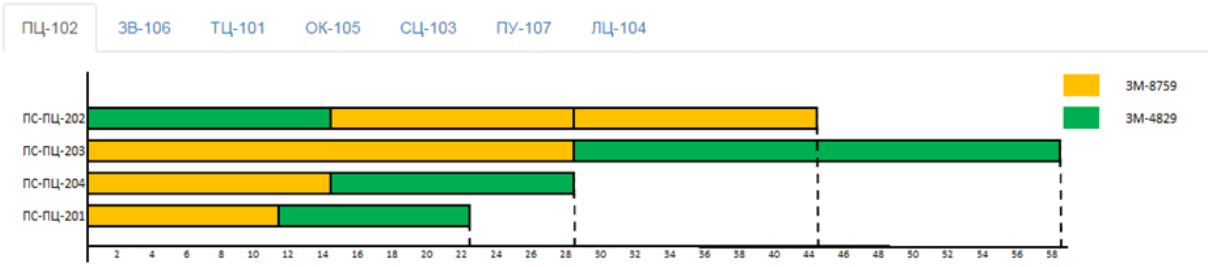


Рис. 4.25. Графік поточного розподілу завдань за пристроями на дільниці

Висновок до розділу 4

У розділі розглянуто розробку інформаційної технології оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк». Наведено опис розробленої архітектури інструментального комплексу ІТОКПДВ, яка представляє собою трирівневу архітектуру, переваги якої було наведено. Розглянуті компоненти інструментального комплексу інформаційної технології та принцип їх функціонування, а також архітектура самого сервера застосувань, яка представляє собою багатошарову архітектуру. Наведені пакети та класи інструментального комплексу ІТОКПДВ з описом основних функцій з їх параметрами та значеннями, розроблена база даних з використанням реляційної бази даних PostgreSQL. Наведено обґрунтування вибору технологій розробки, до яких увійшли наступні технології: високорівнева мова

програмування Java, реляційна база даних PostgreSQL, платформа Java Enterprise Edition, Spring MVC Framework, бібліотека об’єктно-реляційного відображення Hibernate.

Представлено рішення з алгоритмічного забезпечення інформаційної технології ОКПДВ, що демонструє опис її розробки. У рамках розробки інформаційної технології було вирішено задачу формування узагальненої технологічної карти проходження завдань по виробничих дільницях, диспетчеризації складання розкладів виконання завдань на окремих дільницях за узагальненою технологічною картою.

Виявлені особливості побудови узагальнених графів; розглянуті приклади, що ілюструють часткові випадки процесу побудови узагальнених графів. Розроблено принципову схему алгоритму побудови узагальненої технологічної карти, а на її основі – детальний алгоритм; описані складові цього алгоритму – процедури і функції. Розроблено алгоритм диспетчеризації – алгоритм складання розкладу, що базується на основі узагальненого графа.

Наведено короткий огляд програмної реалізації інструментального комплексу ІТОКПДВ.

Після впровадження ІТОКПДВ було проведено аналіз процесу планування виробництва до впровадження та після нього, результати порівняння показали, що після впровадження ІТОКПДВ час на складання календарних планів виробництва зменшився, якість складених планів покращилась, за рахунок виконання замовлень точно в строк та час виконання замовлень зменшився на 5-10%.

ВИСНОВКИ

У *першому розділі* наведено огляд та застосування систем планування і контролю виробництва для організації рівномірної, ритмічної взаємоузгодженої роботи всіх виробничих підрозділів підприємства. Наведено класифікацію різних типів організації виробництва та детальний опис характеристики дрібносерійного виробництва. Розглянуті різні методології та парадигми з управління виробництвом, наведена порівняльна характеристика наступних методологій: Ощадне виробництво, Швидкореагуюче виробництво та Активне виробництво. З урахуванням особливостей дрібносерійного типу виробництва та поставленої мети розробка ІТОКПДВ буде виконуватися в межах виробничої концепції «точно в строк», яка відноситься до методології LM та може за необхідності забезпечити гнучку перебудову виробництва.

Детально розглядається зміст, мета, функції та завдання оперативного управління основним виробництвом, основною складовою якого є оперативно-календарне планування та диспетчеризація, для яких і розробляється інформаційна технологія.

Представлено огляд програмних систем для планування роботи та управління виробництвом. ІТОКПДВ, розробка якої розглядається у даній роботі, є частково інтегрованою системою, яка загалом відноситься до класу ERP-систем, але має у своєму складі елементи APS та MES-систем.

Було показано, що важливою складовою інформаційної технології ОКПДВ є реалізація моделей достатньо складної структури, зокрема моделей теорії розкладів. Наведено класифікацію задач теорії розкладів та описані їх складові і умови вирішення. Розглянуто методи вирішення задач теорії розкладів та проведений детальний аналіз використання цих методів для реалізації схожих задач. У роботі наведено огляд результатів застосування найбільш розповсюджених методів розв’язання задач теорії розкладів. Описано застосування методології побудови ПДС-алгоритмів для важкорозв’язуваних задач комбінаторної оптимізації для розробки алгоритмів складання розкладів.

При проведенні аналізу літературних джерел було виявлено роботи, що пропонують різні підходи до розв’язання досліджуваних задач теорії розкладів.

Наведено вимоги до інформаційної технології, яка розробляється у даній роботі.

У *другому розділі* наведено результати дослідження задач визначення максимально пізнього моменту початку виконання в допустимому розкладі завдань із спільним директивним строком паралельними ідентичними пристроями та пристроями різної продуктивності. Для задачі складання розкладів роботи паралельних ідентичних пристроїв розроблено множину операцій обміну завданнями між пристроями, на основі достатніх умов оптимальності розроблено поліноміальна складова ПДС-алгоритму вирішення задачі, що використовує цю множину операцій обміну. При дослідженні властивостей задачі календарного планування виконання завдань із спільним жорстким директивним строком паралельними пристроями різної продуктивності з метою максимізації моменту запуску пристроїв за умови, що усі завдання не запізнюються, сформульована допоміжна оптимізаційна задача, за результатами якої визначені достатні умови оптимальності розкладів. На основі яких розроблено поліноміальна складова ПДС-алгоритму вирішення задачі, що використовує розроблену множину операцій обміну завданнями між пристроями, які дозволяють послідовно покращувати значення критерію.

Для перевірки ефективності розроблених алгоритмів була проведена серія експериментів. Під час яких на вхід алгоритму подавалися серії задач різних типів. Також при цьому були досліджені різні стратегії пошуку операцій обміну (пошук першого допустимого «простого» обміну (*simple/first*), пошук найкращого з можливих «простих» обмінів (*simple/best*), пошук найкращого з можливих «складних» обмінів (*complex/best*)).

На підставі результатів експериментів отримані такі висновки: найкращі результати демонструє стратегія *complex/best* на коротких завданнях незалежно від розсіювання (100-відсоткове попадання в оптимум); але, в той же час, стратегія *complex/best* є найповільнішою з розглянутих, тому може бути

неприйнятною при потребі швидко розв’язати велику кількість задач; для задач з короткими завданнями усі стратегії пошуку операцій обміну (для усіх задач) дають результати, близькі до оптимуму (відсоток відхилення коливається від 10^{-7} до 10^{-5}; як і очікувалось, збільшення розсіювання тривалостей завдань скорочує час, необхідний для пошуку розв’язку, це особливо помітно на задачах з короткими завданнями; стратегія *simple/first* дозволяє отримувати розв’язок дуже швидко, але при цьому дає найбільший (проте все ще прийнятний) відсоток відхилення від оптимуму; щодо впливу розподілу тривалостей завдань виявлено наступне: у порівнянні з рівномірним розподілом, стратегія *complex/best* для нормального розподілу працює швидше при великих дисперсіях тривалостей, але при цьому частка знайдених оптимальних розв’язків дещо менша (в середньому 15% проти 8%).

У *третьому розділі* наведено дослідження двох задач складання розкладів виконання завдань паралельними пристроями. Одна з них, задача складання розкладу виконання завдань паралельними пристроями однакової продуктивності з метою мінімізації максимуму відхилення моментів завершення завдань пристроями від загального директивного строку, а інша – задача складання розкладу для пристроїв різної продуктивності з метою побудови розкладу з максимально рівномірним розподілом завдань між пристроями. Розглянуті задачі є близькі за змістом. Проведене дослідження цих властивостей задач, в результаті яких сформульовані достатні умови оптимальності розкладів. При цьому, при досліджені властивостей задач календарного планування виконання завдань пристроями різної продуктивності з метою побудови розкладу з максимально рівномірним розподілом завдань між пристроями сформульована допоміжна оптимізаційна задача, за результатами якої визначена одна з достатніх умов оптимальності розкладів. Розроблено алгоритм вирішення допоміжної оптимізаційної задачі. З урахуванням достатніх умов оптимальності розкладів визначено множину операцій обміну, які дозволяють послідовно покращувати значення критерію. На основі достатніх умов оптимальності та

розробленої множини операцій обміну побудовані поліноміальні складові ПДС-алгоритмів вирішення поставлених задач.

Для перевірки ефективності розроблених алгоритмів була проведена серія експериментів. На підставі результатів експериментів отримані такі висновки: найкращі результати демонструє стратегія *complex/best* на коротких завданнях незалежно від розсіювання (100-відсоткове попадання в оптимум), але, в той же час, стратегія *complex/best* є найповільнішою з розглянутих; для задач з короткими завданнями усі стратегії пошуку операцій обміну (для усіх задач) дають результати, близькі до оптимуму (відсоток відхилення коливається від 10^{-7} до 10^{-6}; як і очікувалось, збільшення розсіювання тривалостей завдань скорочує час, необхідний для пошуку розв’язку, це особливо помітно на задачах з короткими завданнями; стратегія *simple/first* дозволяє отримувати розв’язок дуже швидко, але при цьому дає найбільший (проте все ще прийнятний) відсоток відхилення від оптимуму (для коротких та середніх тривалостей відхилення не перевищує 10%).

У *четвертому розділі* розглянуто розробку інформаційної технології оперативно-календарного планування дрібносерійного виробництва за концепцією «точно в строк». Наведено опис розробленої архітектури ІТОКПДВ, яка представляє собою трирівневу архітектуру, переваги якої було наведено. Розглянуті компоненти інформаційної технології та принцип їх функціонування, а також архітектура сервера Web-застосування, яка представляє собою багатошарову архітектуру. Наведені пакети та класи інформаційної технології ОКПДВ з описом основних функцій з їх параметрами та значеннями, розроблена база даних інформаційної технології з використанням реляційної бази даних PostgreSQL. Наведено обґрунтування вибору технологій розробки.

Представлено рішення з алгоритмічного забезпечення ІТОКПДВ. У рамках розробки інформаційної технології було вирішено задачу формування узагальненої технологічної карти проходження завдань по виробничих дільницях та диспетчеризації складання розкладів виконання завдань на окремих дільницях за узагальненою технологічною картою. Виявлені особливості

побудови узагальнених графів; розглянуті приклади, що ілюструють часткові випадки процесу побудови узагальнених графів.

Розроблено принципову схему алгоритму побудови узагальненої технологічної карти, а на її основі – детальний алгоритм; описані складові цього алгоритму – процедури і функції. Розроблено алгоритм диспетчеризації – алгоритм складання розкладу, що базується на основі узагальненого графа.

Наведено короткий огляд програмної реалізації інформаційної технології для оперативно-календарного планування дрібносерійного виробництва.

Після впровадження інформаційної технології ОКПДВ було проведено аналіз процесу планування виробництва до впровадження та після нього, результати порівняння показали, що після впровадження інформаційної технології ОКПДВ час на складання календарних планів виробництва зменшився, якість складених планів покращилась, за рахунок виконання замовлень точно в строк та час виконання замовлень зменшився на 5-10%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сперкач М.О. Поліноміальна складова ПДС-алгоритму розв'язання однієї задачі теорії розкладів / О.А. Павлов, О.Г. Жданова, О.Б. Місюра, М.О. Сперкач // Технологический аудит и резервы производства, 2013. — №6/3 (14). — С.47-52.

2. Сперкач М.О. Задача составления допустимого расписания с максимально поздним моментом запуска выполнения идентичными параллельными приборами работ с общим директивным сроком / А.А. Павлов, М.О. Сперкач, Е.Г. Жданова // Вісник НТУУ "КПІ". Серія «Інформатика, управління та обчислювальна техніка». – К.: "ВЕК+", 2014. – №61 – С.93-102.

3. Сперкач М.О. Выполнение заданий с общим директивным сроком параллельными приборами по критериям оптимальности: минимизация суммарного опережения относительно директивного строка и максимизация момента запуска заданий на выполнение / А.А. Павлов, М.О. Сперкач // Вісник НТУУ "КПІ". Серія «Інформатика, управління та обчислювальна техніка». – К.: "ВЕК+", 2015. – №62 – С.89-92.

4. Сперкач М.О. Субоптимальный полиномиальный алгоритм решения одного класса многоэтапных сетевых задач календарного планирования / А.А. Павлов, М.О. Сперкач, Е.А. Халус // Вісник НТУУ "КПІ". Серія «Інформатика, управління та обчислювальна техніка». – К.: "ВЕК+", 2012. – №57. – С. 51–55.

5. Сперкач М.О. Составление расписания выполнения работ параллельными приборами с целью минимизации максимального отклонения от директивного срока / А.А. Павлов, Е.Г. Жданова, М.О. Сперкач // Вісник ХНТ ім. В.Н. Каразіна. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – Х.: 2015. – №1156 – С.92-106.

6. Сперкач М.О. Задача визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним терміном паралельними пристроями різної продуктивності / М.О. Сперкач // Вісник НТУУ "КПІ". Серія «Інформатика, управління та обчислювальна техніка». – К.: "ВЕК+", 2015. – №63 – С.12-18.

7. Сперкач М.О. Задача складання розкладу виконання завдань паралельними приладами з метою мінімізації максимуму відхилення від директивного терміну моментів завершення приладами усіх завдань / О.А. Павлов, М.О. Сперкач, О.Г. Жданова // Математичне та комп'ютерне моделювання. Серія «Технічні науки». – Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2014. – № 10. – с. 148 – 158.

8. Сперкач М.О. Четырехуровневая модель планирования, принятия решений и оперативного управления в сетевых системах с ограниченными ресурсами / А.А. Павлов, М.О. Сперкач, Е.Б. Мисюра, Т.Н. Лисецкий // Вісник НТУУ "КПІ". Серія «Інформатика, управління та обчислювальна техніка». – К.: "ВЕК+", 2013. – №58 – С. 13.

9. Сперкач М.О. Результирующая формализация первого уровня трехуровневой модели оперативного планирования и принятия решений по критерию минимизации суммарного опережения директивных сроков / А.А. Павлов, М.О. Сперкач, Е.Б. Мисюра, Е.А. Халус, Г.А. Аракелян // Вісник НТУУ "КПІ". Серія «Інформатика, управління та обчислювальна техніка». – К.: "ВЕК+", 2012. – №56. – С. 56–57.

10. Сперкач М.О. Складання розкладу виконання завдань паралельними пристроями різної продуктивності з метою максимально рівномірного завантаження пристроїв / М.О. Сперкач, О.Г. Жданова // Вісник ХНТ ім. В.Н. Каразіна. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – Х.: 2015. – №1156 – С.92-106.

11. Зак Ю.А. Прикладные задачи теории расписаний и маршрутизации перевозок / Ю.А. Зак. – М.: Книжный дом «ЛИБРОКОМ», 2012. – 394 с.

12. Танаев В.С. Введение в теорию расписаний / В.С. Танаев, В.В.Шкурба. – М.: Наука, 1975. – 256 с.

13. Алесинская Т.В. Основы логистики. Функциональные области логистического управления. Часть 3. / Т.В. Алесинская. – Таганрог: Изд-во ТТИ ЮФУ, 2010. – 116 с.

14. Герасимчук В.Г. Стратегічне управління підприємством. Графічне моделювання: Навч. посіб. / В.Г. Герасимчук — К: КНЕУ, 2000. — 360 с.

15. Лузин А.Е. Постфордизм – три ключевые производственные парадигмы нового столетия [Електронний ресурс]/ А.Е. Лузин, Ю.В. Бабанова // Менеджмент в России и за рубежом. – 2013. – №6. – Режим доступу до ресурсу: http://qrmrussia.ru/index.php/publications/22-postfordizm-tri-klyuchevye-proizvodstvennye- paradigmy-novogo-stoletiya.

16. Кузьмин А.М. Метод "Бережливое производство" и другие методы поиска идей и создания инноваций [Електронний ресурс] / А.М. Кузьмин. – Режим доступу до ресурсу: http://www.inventech.ru/pub/methods/metod-0009/.

17. Сури Р. Время — деньги. Конкурентное преимущество быстрореагирующего производства = It's about Time: The Competitibe Advantage of Quick Response Manufacturing / Р. Сури. – Пер.с англ. – М.: Бином. Лаборатория знаний, 2012. – 326 с.

18. Кузин Б.И. Организация и оперативно-календарное планирование машиностроительного производства в АСУП / Б.И. Кузин, В.А. Дуболазов. – Л.: Изд-во Ленингр. ун-та, 1978. – 240 с.

19. Звягінцев Ю.Е. Оперативне планування та організація ритмічної роботи на промислових підприємствах / Ю.Є. Звягінцев. – К.: Техніка, 2002. – 159 с.

20. Shaw C. Manufacturing Planning and Execution Software Interfaces / C. Shaw // Journal of Manufacturing Systems. – 2000. – Vol. 19, №1. – P. 1–17.

21. Мизюн В.А. Управление производственными системами и процессами [Електронне видання] / В.А. Мизюн. – Самара: Издательство СНЦ РАН, 2012. – 211 с. – Режим доступу до ресурсу: http://www.cfin.ru/management/manufact/ manufacturing_sys-03.shtml .

22. Антонов А.Н. Основы современной организации производства / А.Н. Антонов, Л.С. Морозова. – М.: Дело и сервис, 2004. – 578 с.

23. Горнев В.Ф. Оперативное управление в ГПС / В.Ф. Горнев, В.В. Емельянов, М.В. Овсянников. – М.: Машиностроение, 1990. – 256 с.

24. Жукова И.С. Проблемы организации производства в условиях смены технологических укладов / И.С. Жукова // Теоретические основы и практика организации производства: Юбилейный сб. науч. трудов. — Воронеж : ВГТУ , 2010. – С. 13 - 20.

25. Ковалев В.В. Анализ деятельности предприятия: учеб. [Текст] / В.В. Ковалев, О.Н. Волкова. — М.: издательство «Проспект», 2004. – 424 с.

26. Мизюн В. А. Модель конкурентоспособного производства / В. А. Мизюн // Аудит и финансовый анализ. – 2009. – № 5. – С. 314 – 344.

27. Мильнер Б.З. Теория организации. / Б.З. Мильнер. – 6-е изд. (перераб. и доп.). – М.: ИНФРА-М, 2008. – 797 с.

28. Мыльник В.В. Исследование систем управления / В.В. Мыльник, Б.П. Титаренко, В.А. Волочиенко. – 4-е издание. – М.: Академический проект, 2006. – 352 с.

29. Родионова В.Н. Понятие и механизм синхронизации производственных процессов / В.Н. Родионова // Организатор производства. – М.: Экономика и финансы, 2010. – № 3. – С.15–18.

30. Конвей Р.В. Теория рас писаний / Конвей Р.В., Максвелл В.Л., Миллер Л.В. – М.: Физматгиз, Наука, 1975. – 359 с.

31. Шкурба В.В. и др. Задачи календарного планирования и методы их решения / В.В. Шкурба и др. – Киев: Наукова думка, 1966. – 155 с.

32. Domschke W. Produktionsplanung. Ablauforganisatorische Aspekte / Domschke W., Sholl A., Vob S. – Berlin.: Heildelberg, Springer Verlag, 2005. – 456 S.

33. Weng M. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective [Текст] / M. Weng, X.J. Lu, H. Ren // International Journal of Production Economics. – 2001. – Vol. 70. – P. 215–226.

34. Фролов Евгений. Оперативное планирование производства / Евгений Фролов // «Директор информационной службы». – 2013. – № 05. – С.24–27.

35. Bidgoli H The Internet Encyclopedia [Текст] / H. Bidgoli // John Wiley & Sons. – 2004. – Vol. 1. – P. 707.

36. О'Лири Дэниел. ERP-системы. Современное планирование и управление ресурсами предприятия / Дэниел О'Лири // "Вершина" – 2004 – 272с.

37. Колесов А. Gartner о мировом рынке средних ERP-решений [Текст] / А. Колесов // PC Week/RE. – 2010. – № 29–30. – С. 683–684.

38. Классификация HRM-систем [Електронний ресурс] –Режим доступу до ресурсу: http://www.molga.ru/tadviser/page_three.php#.

39. Система управления складом (WMS) [Електронний ресурс] –Режим доступу до ресурсу: http://wms.korusconsulting.ru/decisions/mw/.

40. SCM (Supply Chain Management) – управление цепочками поставок (управление запасами) [Електронний ресурс] – Режим доступу до ресурсу: http://www.tadviser.ru/index.php/Статья:SCM_(Supply_Chain_Management) – управление цепочками поставок (управление_запасами).

41. CRM-системы [Електронний ресурс] – Режим доступу до ресурсу: http://www.norbit.ru/products/groups/188.html.

42. Что такое PLM? [Електронний ресурс] – Режим доступу до ресурсу: http://www.calscenter.ru/?page_id=215.

43. EAM – система, ориентированная на сокращение затрат, связанных с обслуживанием оборудования [Електронний ресурс] – Режим доступу до ресурсу: http://pro-spo.ru/erp/2069-eam.

44. Senthilkumar P. Review of Single Machine Scheduling Problem with Uniform Parallel Machines / P. Senthilkumar, S. Narayanan // Intelligent Information Management. – 2010. – №2. – P. 457–474.

45. Подчасова Т.П., Португал В.М. и др. Эвристические методы календарного планирования / Т.П.Подчасова, В.М. Португал и др. – Киев.: Техника, 1980. – 140 с.

46. Pinedo M.L. Scheduling. Theory, Algorithms and Systems / Michael L. Pinedo. – Springer, 2008. – 671 p.

47. Зак Ю.А. Оптимальное распределение технологических операцій на сборочном контейнере / Ю.А. Зак // Кибернетика. – К., 1990. – №4. – с.45-54.

48. Танаев В.С. Теория расписаний. Групповые технологии / Танаев В.С., Ковалев М.Я., Шафранский Я.М. – Минск.: Институт технической кибернетики НАН Беларуси, 1998. – 289 с.

49. Танаев В.С. Теория расписаний. Одностадийные системы / Танаев В.С., Гордон В.С., Шафранский Я.М. – М.: Физматгиз, Наука, 1984. – 384 с.

50. Танаев В.С. Теория расписаний. Многостадийные системы / Танаев В.С., Сотсков Ю.Н., Струсевич В.А. – М.: Наука, 1989. – 328 с.

51. Brian G.R. Sequencing production on parallel machines with two magnitudes of sequence-dependent setup cost [Текст] / G.R. Brian, S.M. Gilbert // Journal of Manufacturing and Operations Management. – 1990. – Vol. 3 – P. 24–52.

52. Tang C.S. Scheduling batches on parallel machines with major and minor set-ups [Текст] / C.S. Tang // European Journal of Operations Research. – 1990. – Vol. 46 – P. 28–37.

53. Лазарев А.А. Теория расписаний. Минимизация суммарного запаздывания для одного прибора / А.А. Лазарев, Е.Р. Графов. – М.: РАН Вычислительный центр А.А. Дородницына, 2004. – 150 с.

54. Зак Ю.А. Построение допустимих и оптимальних расписаний выполнения работ на одной машине / Ю.А. Зак // Кибернетика. – К., 2011. – №6. – с.47-59.

55. Carlier J. The one-machine sequencing problem / J. Carlier // European Journal of Operational Research. – 1982. – №11. – S.24-47.

56. Brucker P. Scheduling Algorithms / P. Brucker. – Springer-Verlag, Berlin, Heidelberg und New York, 2001. – P. 107 – 155.

57. Зак Ю.А. Определение порядка выполнения независимых операций на параллельных машинах / Ю.А. Зак // Изв. АН СССР. Техническая кибернетика. – К., 1969. – №2. – С. 15–20.

58. Kramer A. A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems [Електронний ресурс]/ A. Kramer, A. Subramanian. – Brazil, Working Paper UFPB, 2015. – Режим доступу до ресурсу: journals/corr/KramerS15.

59. Brucker P. Scheduling Algorithms / P. Brucker. – Springer-Verlag, Berlin, Heidelberg und New York. – 4th ed. – 2004. – P. 87 – 102.

60. Зак Ю.А. Решение обобщенной задачи Джонсона с ограничениями на сроки выполнения заданий и времена работы машин. Ч.1. Точные методы решения / Ю.А. Зак // Проблемы управления. – М., 2010. – №3. – С. 17–25.

61. Зак Ю.А. Некоторые свойства задач теории расписаний / Ю.А. Зак // Автоматика и телемеханика. – М., 1978. – №1. – С. 123–132.

62. Brucker P. A new lower bound for the job-shop scheduling problem / P. Brucker, B. Jurisch // European Journal of Operational Research. – 1993. – №64 (2). – P. 156 – 167.

63. Zack Yu. Mathematisches modell und Algorithmen der Termin- und Reihefolgeplanung (in Deutsch) [Електронний ресурс] / Yu. Zack, S. Rotin. – 2004. – Режим доступу до ресурсу: http://www.optimorum.de/doc/J_II_Cmax.pdf.

64. Ma Y. A Survey of Scheduling with Deterministic Machine Availability Constraints [Текст] / Y. Ma, C. Chengbin, Z. Chunrong // Computers & Industrial Engineering. – 2010. – Vol. 50 – P. 199–211.

65. Allahverdi A. Two-machine ordered flowshop scheduling under random breakdowns [Текст] / A. Allahverdi, J. Mittenthal // Mathematical and computer modelling. – 1994. – 20(2). – P. 9–17.

66. Красовский Д.В. Алгоритмы решения задачи составления оптимального расписания без перерываний: дис. канд. фіз.-мат. наук: 05.13.18 / Красовский Д.В.– М.: МФТИ, 2007. – 109 с.

67. Корбут А.А. Дискретное программирование / А.А. Корбут, Ю.Ю. Финкельштейн. – М.: Наука, Физматгиз, 1969. – 368 с.

68. Krelle W. Ganzzahlige Programmierungen. Theorie und Anwendungen in der Praxis / W. Krelle // Unternehmenforschung, 1958. – №2. – S. 161 – 175.

69. Zimmermann H.-J. Netzplantechnik / H.-J. Zimmermann. – Verlag Walter de Gruyter, Berlin/New York, 1971. – S. 156.

70. Ore O. Theory of graphs, American mathematical Society / O. Ore. – 1962. – (Colloquium Publications). – Vol. 38. – 270 p.

71. Сергиенко И.В. Задачи дискретной оптимизации. Проблемы, методы решения, исследования / И.В. Сергиенко, В.П. Шило. – К.: Наукова думка, 2003. – 260 с.

72. Жиглявский А.А. Методы поиска глобального экстремума / А.А. Жиглявский, А.Г. Жилинскас. – М.: Наука, 1991. – 205 с.

73. Гурин Л.С. Задачи и методы оптимального распределения ресурсов / Л.С. Гурин, Я.С. Дымарский, А.Д. Меркулов. – М.: Сов. Радио, 1968. – 464 с.

74. Зак Ю.А. Методы оптимизации и их применение в целлюлозно-бумажной промышленности / Ю.А. Зак, Р.М. Рейдман, А.А. Рувинский. – М.: Лесная промышленость, 1973. – 248 с.

75. Zack Yu. A. Methods of Multiextremal Optimization under Constraints for Separably Quasimonotone Functions / Yu Zack // Journal of Computer and Systems Sciences International, 2011. – vol.50, №3. – P. 37 – 391.

76. Glover F. Tabu Search, Part II / F. Glover // ORSA Journal on Computing, 1990. – vol.2, №1. – P. 4-32.

77. Емельянов В.В. Теория и практика эволюционного моделирования / В.В. Емельянов, В. В. Курейчик. – М.: Физматлит, Наука, 2003. – 432 с.

78. Курейчик В.М. Генетические алгоритмы: Монография / В.М. Курейчик. – Таганрог: Изд.ТРТУ, 1998. – 242 с.

79. Cleveland G.A. Using genetic algorithms to schedule flow shop releases / G.A. Cleveland Smith S.F.// In. Proceeding of theThird International Conference on Genetic Algorithms. Morgan Kaufmann Publishers. – San Mateo,California. – 1989. – P. 160-169.

80. Glover F. Tabu Search, Part I / F. Glover // ORSA Journal on Computing. – 1989 – Vol. 1, No 3. – P. 190-206.

81. Glover F. Tabu Search, Part II / F. Glover // ORSA Journal on Computing. – 1990 – Vol. 2, No 1. – P. 4-32.

82. Nissen Volker. Einfuhrung in Evolutionare Algorithmen. / Volker Nissen // Optimierung nach dem Vorbild der Evolution. – Vieweg, Munchen, 1997. –P. 345.

83. Goldberg David E. Genetic Algorithms in Search / David E. Goldberg // Optimization, and Machine Learning. – Adison-Wesley, 1998. – 403 s.

84. Michalewicz Z. Heuristic methods for evolutionary computation tech-niques / Z. Michalewicz //Journal of Heuristics. – 1995. - №1. – P. 177-206.

85. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs / Z. Michalewicz. – Springer, Berlin, 1999. – P. 67.

86. Метод комбинаторных евристик для решения комбинаторных задач упорядочения и распределения ресурсов / Д.И. Батищев, Э.Д. Гудман, И.П. Норенков, М.Х. Прилуцкий. – М.: Информационные технологи, 1997. – С. 29-32.

87. Hundal T.S. An extension of Palmer's heuristic for the flow-shop scheduling problem / T.S. Hundal, J. Rajgopal // International Journal of Produktion Reasearch. – 1988. – №26ю – P. 1119 – 1124.

88. Gupta J.N.D. A functional heuristic algorithm for the flop-shop scheduling problem / J.N.D. Gupta // Operational Research Quartrrly. – 1971. – №2. – P. 39-47.

89. Cambell H.G. A heuristic algorithm for the n job, m machine sequencing problem / H.G. Cambell, R.A. Dudek, M.L. Smith // Management Science. – 1970 – №16. – P. 630-637.

90. Dannenbring D.G. A evoluation of flow shop sequencing heuristics / D.G. Dannenbring // Management Scine. – 1977. – №23. – P. 1174-1182.

91. Ho J.C. A new heuristic algorithm for the n-job, M-machine problem / J.C. Ho, Y.-L. Chang // European Journal of Operational Research – 1991. – №52. – P. 194-202.

92. Ishibuchi H., Misaki S., Tanaka H. Modived simulated annealing algorithms for the flow shop sequencing problem / H. Ishibuchi, S. Misaki, H. Tanaka // European Journal of Operational Research. – 1995. – №81. – P. 388-398.

93. Monma C.L. Analysis of heuristics for preemptive parallel machine scheduling with batch setup times [Текст] / C.L. Monma, C.N. Potts // Operations Research. – 1993. – Vol. 41. – P. 981–993.

94. Lee H. A hybrid bounding procedure for the workload allocation problem on parallel unrelated machines with setups [Текст] / H. Lee, M. Guignard // Journal of the Operational Research Society. – 1996. – Vol. 47. – P. 1247–1261.

95. Lawler E.L. Branch-and-bound methods: A Survey / E.L. Lawler, D.E. Wood // Oper. Res. – 1966. – Vol.4, §14. – P. 252 – 260.

96. Land A.H. An automatic method of solving discrete programming problems / A.H. Land, A.G. Doig // Econometrica. – 1960. – Vol.28. – P. 497–520.

97. Беллман Р. Динамическое программирование и современная теория управления = Dynamic Programming and Modern Control Theory / Р. Беллман, Р.Калаба. – Пер. с англ. – М.: Наука, 1969. – 119 с.

98. Лежнев А.В. Динамическое программирование в экономических задачах / А.В. Лежнев. – Бином. Лаборатория знаний, 2010. – 176 с.

99. Михалевич В.С. Вычислительные методы исследований и проектирования сложных систем / В.С. Михалевич, В.Л. Волкович. – М.: Фитматгизм, Наука, 1982. – 287 с.

100. Эванс Э. Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем. : Пер. с англ. / Э.Эванс. – М.: ООО «И.Д. Вильямс», 2014. – 448 с.

101. Webster S. Dynamic programming algorithms for scheduling parallel machines with family setup times [Текст] / S. Webster, M. Azizoglu // Computers & Operations Research. – 2001. – Vol. 28. – P. 127–137.

102. Згуровский М.З. Принятие решений в сетевых системах с ограниченными ресурсами [Текст]: монография / М.З. Згуровский, А.А. Павлов. – К.: Наукова думка, 2010.– 573 с.

103. Красовский Д.В. Алгоритмы решения минимаксной задачи составления расписания / Д.В. Красовский, М.Г. Фуругян // Известия РАН. Теория и системы управления. – 2008. – №5. – С. 69–74.

104. Гончаров Е.Н. Вероятностный поиск с запретами для дискретных задач безусловной оптимизации / Е.Н. Гончаров, Ю.А. Кочетов // Дискретный анализ и исследования операцій. Сер.2. – 2002. – Т.9, №2. — С.13—30.

105. Кочетов Ю.А. Локальный поиск с чередующимися окрестностями / Ю. Кочетов, Н. Младенович, П. Хансен // Дискретный анализ и исследования операций. Серия 2. – 2003. – Т.10, №1. – С. 11–44.

106. Кочетов Ю.А. Использование чередующихся окрестностей для приближенного решения задачи календарного планирования с ограниченными ресурсами / Ю.А. Кочетов, А.А. Столяр // Дискретный анализ и исследования операций. Серия 2. – 2003. – Т.10, №2. – С. 29–56.

107. Алексеев О.Г. Комплексное применение методов дискретной оптимизации / О.Г. Алексеев. – М.: Наука, 1986.

108. Штовба С.Д. Муравьиные алгоритмы / С.Д. Штовба // ExponentaPro. Математика в приложениях. – 2003. – №4(4). – С. 70–75.

109. Lin Y. Unrelated Parallel Machines Scheduling Problem Using an Ant Colony Optimization Approach [Текст] / Y. Lin, H. Hsieh, F. Hsieh // World Academy of Science, Engineering & Technologyю – 2012.

110. Senthil Kumar. Ant Colony Approach for Makespan Minimization on Unrelated Parallel Machines [Електронний ресурс] / K.M. Senthil Kumar, V. Selladurai, K. Raja, K. Elangovan // International Journal of Engineering Science & Technology. – 2011. – P. 3113 - 3120. – Режим доступу до ресурсу: https://www.researchgate.net/publication/267717836_Ant_Colony_Approach_for_Makespan_Minimization_on_Unrelated_Parallel_Machines.

111. Glover F., Laguna M. Chapter 3: Tabu search/Ed. R. Colin Reeves. Modern Heuristics Techniques for Combinatorial Problems. Oxford: Blackwell Scientific Publications, 1993. P. 70-150.

112. Raghavan R. Probabilistic Contruction of Deterministic Algorithms: Approximating Packing Integer Programs / R. Raghavan // J. Computer and System Sciences. – 1988. – Vol. 37. – P. 130–143.

113. Костенко В.А. Синтез структур вычислительных систем реального времени с использованием генетических алгоритмов / В.А. Костенко, Р.Л. Смелянский, А.Г. Трекин // Программирование. – 2000. –№5 – С. 63–72.

114. Vacher J. P. Genetic algorithms in a multi-agent system. In Intelligence and Systems [Текст] / J. P. Vacher., T. Galinho, F. Lesage, A. Cardon // Proceedings, IEEE International Joint Symposia. – 1998. – P. 17–26.

115. Moon C. Integrated machine tool selection and operation sequencing with capacity and precedence constraints using genetic algorithm [Текст] / C. Moon, M. Lee, Y. Seo, Y.H. Lee // Computers & industrial engineering. – 2002. – 43(3). – P. 605–621.

116. A genetic algorithm approach for minimizing total tardiness in parallel machine scheduling problems [Текст] / Tufan Demiral, Nihan Cetin Demirel, Belgin Tasdelen, Vildan Ozkir // Proceedings of the World Congress on Engineering. – 2011. – P. 1190 - 1193.

117. Chaudhry Imran Ali. Minimizing flow time for the worker assignment problem in identical parallel machine models using GA [Текст] / Imran Ali Chaudhry // International Journal of Advanced Manufacturing Technology. – 2010. – P. 747 - 760.

118. Chen Jeng-Fung. Unrelated parallel-machine scheduling to minimize total weighted completion time [Електронний ресурс] / Jeng-Fung Chen // Springer-Verlag. – London, 2013. – P.8-25. – Режим доступу до ресурсу: https://www.infona.pl/ resource/bwmeta1.element.springer-doi-10_1007-S10845-013-0842-Y.

119. Alcan Pelin. An Application with Non-identical Parallel Machines using Genetic Algorithm with the Help of Fuzzy Logic [Електронний ресурс] / Pelin Alcan // Proceedings of the World Congress on Engineering. – 2011. – Режим доступу до ресурсу: http://www.iaeng.org/publication/WCE2011/ WCE2011_pp1166-1169.pdf.

120. Алгоритмы: построение и анализ = Introduction to Algorithms / Кормен Т.Х. [та ін.]. – 2-е издание (пер. с англ.). – М.: Издательский дом «Вильямс», 2007. – 1296 с.

121. Vairam S. Parallel machine shop scheduling using memetic algorithm [Текст] / S. Vairam, V. Selladurai // Applied Mechanics and Materials, Vol. 573. – 2014. – P. 362-367.

122. Vairam S. Parallel machine shop scheduling using memetic algorithm [Текст] / S. Vairam, V. Selladurai // Applied Mechanics and Materials, Vol. 573. – 2014. – P. 362-367.

123. Guo Peng. Parallel machine scheduling with step deteriorating jobs and setup times by a hybrid discrete cuckoo search algorithm [Електронний ресурс] / Peng Guo, Wenming Cheng, Yi Wang // Engineering Optimization. – 2013. – P. 1-22. – Режим доступу до ресурсу: http://arxiv.org/pdf/1309.1453v1.pdf.

124. Parallel-Machine Scheduling Problem under the Job Rejection Constraint [Текст] / Weidong Li, Zhiban Chen, Xuejie Zhang, Jianping Li // Frontiers in Algorithmics. – China, 2014. – P.158-159.

125. Caniyilmaz Erdal. An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date [Текст] / Erdal Caniyilmaz, Betul Benli, Mehmet Ilkay // International Journal of Advanced Manufacturing Technology. Springer-Verlag. – London, 2014. – P.9-12.

126. Rustogi Kabir A. Parallel Machine Scheduling: Impact of Adding Extra Machines [Текст] / Kabir Rustogi, Vitaly Strusevich // Operations Research. – 2013. – P. 1243 - 1257.

127. Zou Juan. Minimizing makespan with chain precedence constraints on identical parallel machines. [Текст] / Juan Zou, Yuzhong Zhang, Longchun Wang // Advances in Information Sciences & Service Sciences. Vol. 4, Issue 21. – 2012. – P. 1190 - 1193.

128. Cheng Zhenmin. An approximate algorithm for parallel machine scheduling problem to minimize total completion time [Текст] / Zhenmin Cheng // Computing & Information Systems. – 2010. – P. 187 - 198.

129. Kolahan F. A heuristic algorithm approach for scheduling of multi-criteria unrelated parallel machines [Текст] / F. Kolahan, V. Kayvanfar // World Academy of Science, Engineering & Technology – 2009. – P.102-106.

130. Laarhoven P. Job Shop Scheduling by Simulated Annealing / P. Laarhoven, E. Aarts, J. Lenstra // Operations Research. – 1992. – Vol. 40(1). – P. 113–125.

131. Shen C. Scheduling multiple job problems with guided evolutionary simulated annealing approach / C. Shen, Y. Pao, P. Yip // Proc. First IEEE Conf. on Evolutionary Computations. – Orlando, 1994. – P. 702–706.

132. Lin Shih-Wei. A multi-point simulated annealing heuristic for solving multiple objective unrelated parallel machine scheduling problems [Електронний ресурс] / Shih-Wei Lin, Kuo-Ching Ying // International Journal of Production Research. – 2014. – P. 1 - 41. – Режим доступу до ресурсу: https://www.researchgate.net/publication/269280769_A_multi-point_simulated_annealing_heuristic_for_solving_multiple_objective_unrelated_parallel_machine_scheduling_problems.

133. Zhang Rui. A simulated annealing-based heuristic algorithm for Job Shop scheduling to minimize lateness [Текст] / Rui Zhang // International Journal of Advanced Robotic Systems. – 2013. – P. 91-108.

134. Головкин Б.А. Расчет характеристик и планирования параллельных вычислительных процессов / Б.А. Головкин. – М: Радио и связь, 1983. –272 с.

135. Jeong Suk Jae. Parallel machine scheduling with earliness-tardiness penalties and space limits [Електронний ресурс] / Suk Jae Jeong, Kyung Sup Kim // Springer-Verlag. Volume 37, Issue 7. London, 2008. – P. 793-802. – Режим доступу до ресурсу: http://link.springer.com/article/10.1007%2Fs00170-007-1027-7.

136. Akyol Emine. A Variable Capacity Parallel Machine Scheduling Problem [Текст] / Emine Akyol, Tugba Sarac // Proceedings of the International Conference on Industrial Engine and Operations Management Istanbul, Turkey, 2012. – P. 548-554.

137. Unrelated parallel machines scheduling problem with sequence dependent setup times [Текст] / Vahid Kayvanfar, Amin Aalaei, Mahtab Hosseininia, Mahdi Rajabi // Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management. – 2014. – P. 1794 - 1803.

138. Toksari M.D. Minimizing the earliness/tardiness costs on parallel machine with learning effects and deteriorating jobs: a mixed nonlinear integer programming approach. [Текст] / M.D. Toksari, E. Guner // Springer-Verlag. – London, 2008. – P. 801-807.

139. Cheng Wenming. Variable Neighborhood Search for Parallel Machines Scheduling Problem with Step Deteriorating Jobs [Електронний ресурс] / Wenming Cheng, Peng Guo, Jian Liang // Mathematical Problems in Engineering. – 2012. – P. 1-11. – Режим доступу до ресурсу: http://connection.ebscohost.com/c/articles/87029741/variable-neighborhood-search-parallel-machines-scheduling-problem-step-deteriorating-jobs

140. Senthilkumar K.M. A Hybrid Algorithm Based on PSO and ACO Approach for Solving Combinatorial Fuzzy Unrelated Parallel Machine Scheduling Problem [Текст] / K.M. Senthilkumar, K. Raja // European Journal of Scientific Research. – 2011. – P. 87-104. – Режим доступу до ресурсу: http://connection.ebscohost.com/c/articles/99708438/hybrid-clustering-algorithm-based-fuzzy-c-means-improved-particle-swarm-optimization.

141. Park Y. Scheduling jobs on parallel machines applying neural network and heuristic rules [Текст] / Y. Park, S. Kim, Y.-H. Lee // Computers & Industrial Engineering. – 2000. – Vol. 38. – P. 189–202.

142. Технологічна карта [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Технологічна_карта.

143. Azizoglu M. Tardiness minimization on parallel machines [Текст] / M. Azizoglu, O. Kirca // International Journal of Production Economics. – 1998. – Vol. 55. – P. 163–168.

144. Ghirardi M. Makespan Minimization for Scheduling Unrelated Parallel Machines: A Recovering Beam Search Approach / M. Ghirardi, C.N. Potts // European Journal of Operational Research. – 2005. – Vol. 165, №2. – P. 457–467.

145. Gairing M. A faster combinatorial approximation algorithm for scheduling unrelated parallel machines / M. Gairing, B. Monien, A. Wroclaw // Theoretical Computer Science. – 2007. – P. 156–169.

146. Fanjul-Peyró L. Iterated greedy local search methods for unrelated parallel machine scheduling [Електронний ресурс] / L. Fanjul- Peyró, R. Ruiz. – 2009. – Режим доступу до ресурсу: http://www.upv.es/ deioac/Investigacion/ Paralelas_Fanjul.pdf.

147. Albers S. Online Makespan Minimization with parallel schedules [Електронний ресурс] / S. Albers, M. Hellwing. – Режим доступу до ресурсу: http://arxiv.org/pdf/1304.5625v1.pdf.

148. Helal M. A Tabu Search Algorithm to Minimize the Makespan for the Unrelated Parallel Machines Scheduling Problem with Setup Times / M. Helal, G. Rabadi, A. Al-Salem // International Journal of Operations Research. –2006. – Vol. 3, №3. – P. 182–192.

149. Azizoglu M. On the minimization of total weighted flow time with identical and uniform parallel machines / M. Azizoglu, O. Kirca // European Journal of Operational Research. – 1999– v. 113. – n. 1. – P. 91 – 100.

150. Prabuddha D. Scheduling to Mini-mize Makespan on Unequal Parallel Processors / D. Prabuddha, E. Thomas // Deci-sion Sciences. – 1980 – v. 11. – n. 4. – P. 586-602.

151. Kovalyov M.Y. Uniform Ma-chine Scheduling of Unit-Time Jobs Subject to Resource Constraints / M.Y. Kovalyov, Y.M. Shafransky // Discrete Applied Mathematics. – 1998 – v. 84. – n. 1-3. – P. 253-257.

152. Burkard R.E. A Linear Com-pound Algorithm for Uniform Machine Scheduling / R.E. Burkard, Y. He, H. Kellerer // Computing. – 1998 – v. 61. – n. 1. – P. 1-9.

153. Chekuri C. An Efficient Approximation Algorithm for Minimizing Makespan on Uniformly Re-lated Machines / C. Chekuri, M. Bender // Proceedings of IPCO’99, LNCS. – 1999 – P. 383-393.

154. Chekuri C. An Efficient Approximation Algorithm for Minimizing Makespan on Uniformly Re-lated Machines / C. Chekuri, M. Bender // Journal of Algorithms. – 2001 – v. 41. – n. 2. – P. 212-224.

155. Liao C.J. Makespan Minimization for Two Uniform Parallel Machines / C.J. Liao, C.H. Lin // International Journal of Production Economics. – 2003 – v. 84. – n. 2. – P. 205- 213.

156. Epstein L. Approximation Schemes for Scheduling on Uniformly Rela ted and Identical Parallel Machines / L. Epstein, J. Sgall // Proceedings of 7 th Annual European Sympo-sium on Algorithms. – 1999 – P. 151-162.

157. Koulamas C. Makespan Minimiza-tion on Uniform Parallel Machines with Release Times / C. Koulamas, G. J. Kyparisis // European Journal of Operational Research. – 2004 – v. 157. – n. 1. – P. 262-266.

158. Liao C.J. Makespan Minimization for Multiple Uniform Machines / C.J. Liao, C.H. Lin // Computers & Industrial Engineering. – 2008 – v. 54. – n. 4. – P. 983-992.

159. Shachnai H. Minimizing Makespan and Preemption Costs on a System of Uniform Machines / H. Shachnai, T. Tamir, G. J. Woeginger // Proceedings of the 10th European Symposium on Algorithms. – 2002 – P. 859-871.

160. Павлов А.А. Признаки оптимальности допустимых решений труднорешаемых задач комбинаторной оптимизации / А.А. Павлов// Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2013. – №59. – С.4-12.

161. Сперкач М.О. Множина перестановок завдань як складова ПДС-алгоритму розв’язання однієї задачі теорії розкладів [Текст] / О.Г. Жданова, М.О. Сперкач // «Актуальні проблеми гуманітарних та природничих наук», м. Одеса, 03-04 квітня 2015 р. — Херсон: Видавничий дім "Гельветика", 2015.

162. Сперкач М.О. Сравнительные характеристики задач составления расписаний выполнения заданий с общим директивным сроком параллельными приборами равной производительности по различным критериям оптимальности / А.А. Павлов, Е.Б. Мисюра, М.О. Сперкач // Інформаційні технології як інноваційний шлях розвитку України у XXI столітті: Матеріали I Міжнародній науково-практичній конференції молодих науковців 06–08 грудня 2012 р. – Ужгород: Закарпатський державний університет, 2012. – С. 109–112.

163. Сперкач М.О. Достатні умови оптимальності розкладу задачі визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним терміном паралельними пристроями різної продуктивності / М.О. Сперкач, О.Г. Жданова // Матеріали десятої міжнародної науково-практичної конференції «Математичне та імітаційне моделювання систем МОДС 2015», Київ-Жукин,22-26 червня 2015 р. – м. Чернігів.: ЧДІЕУ. – С.109-112.

164. Сперкач М.О. Задача визначення максимального пізнього моменту початку виконання завдань із спільним жорстким директивним терміном паралельними приладами різної продуктивності / М.О. Сперкач, О.Г. Жданова // Матеріали дев'ятої міжнародної науково-практичної конференції «Математичне та імітаційне моделювання систем МОДС 2014», Київ-Жукин, 23-27 червня 2014 р. – м. Чернігів.: ЧДІЕУ. – С. 108 – 112.

165. Сперкач М.О. Задача визначення максимально пізнього моменту початку виконання в допустимому розкладі завдань із спільним директивним терміном паралельними приладами з різною продуктивністю / М.О. Сперкач, О.Г. Жданова // Матеріали доповідей міжнародної науково-практичної конференції «Актуальні проблеми економіки та управління сучасної України», м. Ужгород, 16-17 травня 2014 р. / За. заг. ред.: М.М. Палінчак, В.П. Приходько. – Ужгород: Видавничий дім «Гельветика», 2014. – С. 291 – 293.

166. Сперкач М.О. Застосування бджолиного алгоритму для розв'язання задачі складання розкладу виконання завдань паралельними ідентичними приладами / М.О. Сперкач, К.М. Новак // Матеріали доповідей міжнародної науково-практичної конференції «Актуальні проблеми економіки та управління сучасної України», м. Ужгород, 16-17 травня 2014 р. / За. заг. ред.: М.М. Палінчак, В.П. Приходько. – Ужгород: Видавничий дім «Гельветика», 2014. – С. 312 – 316.

167. Сперкач М.О. Властивості перестановок ПДС-алгоритму розв'язання задачі складання розкладу виконання завдань паралельними приладами з метою мінімізації максимуму відхилення від директивного терміну моментів

завершення приладами усіх завдань / О.Г. Жданова, Т.О. Морозовський, М.О. Сперкач // Матеріали десятої міжнародної науково-практичної конференції «Математичне та імітаційне моделювання систем МОДС 2015», Київ-Жукин, 22-26 червня 2015 р. – м. Чернігів.: ЧДІЕУ. – С.113-116.

168. Сперкач М.О. Ознаки оптимальності для складання розкладу виконання завдань паралельними пристроями з метою мінімізації максимуму відхилення від директивного терміну моментів завершення пристроями усіх завдань [Електронний ресурс] / О.А. Павлов, М.О. Сперкач, О.Г. Жданова // Матеріали Міжнародної науково-технічної конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем управління організаційно-технічними та технологічними комплексами», 27 листопада 2014 р. – К: НУХТ, 2014 р. – С. 76-78. — Режим доступу: http://nuft.edu.ua/page/view/konferentsii .

169. Сперкач М.О. Складання розкладу виконання завдань паралельними приладами з метою мінімізації максимуму відхилення від директивного терміну моментів завершення приладами усіх завдань / О.А. Павлов, М.О. Сперкач, О.Г. Жданова // Матеріали 21-ї міжнародної конференції з автоматичного управління «Автоматика-2014», м.Київ, 23-27 вересня 2014 р. – Чернігів.: Вид-во НТУУ «КПІ» ВПІ ВПК «Політехніка», 2014. – С. 212 – 214.

170. Сперкач М.О. Достатні умови оптимальності розкладу виконання завдань паралельними пристроями різної продуктивності з максимально рівномірним завантаженням пристроїв / О.Г. Жданова, М.О. Сперкач // Матеріали десятої міжнародної науково-практичної конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту», 25-28 травня 2015 р. – м. Херсон: ХНТУ. – С. 61-63.

171. Сперкач М.О. Формування графа передування робіт багатоетапної задачі складання розкладу з метою мінімізації сумарного випередження директивних термінів [Текст]: матеріали IV Всеукраїнської заочної науково-практичної конференції «Сучасні інформаційні технології», м. Київ, 27 травня

2013 р.: тези / М.О. Сперкач, Т.О. Морозовський, К.О. Лук'яненко / [редкол.: Жданова О.Г. та ін.] – С. 70–73.

172. Сперкач М.О. Побудова узагальненого графа передування робіт багатоетапної задачі календарного планування мінімізації сумарного випередження директивних термінів / О.Г. Жданова, К.О. Лук'яненко, Т.О. Морозовський, М.О. Сперкач // Математичне та імітаційне моделювання систем: матеріали 8-ї Міжнародної науково-практичної конференції МОДС 2013 – Чернігів, 2013. – С. 150-154.

173. Belouadah H. Scheduling identical parallel machines to minimize total weighted completion time / H. Belouadah, C. Potts // Discrete Applied Mathematics. – 1994– v. 48. – n. 3. – P.201 – 218.

174. Guinet A. Scheduling independent jobs on uniform parallelmachines to minimize tardiness criteria / A. Guinet // Journal of Intelligent Manufacturing. – 1995– v. 6. – n. 2. – P. 95–103.

175. McCormic S. Scheduling n Independ-ent Jobs on m Uniform Machines with both Flow Time and Makespan Objectives: A Parametric Analysis / S. McCormic, M. Inedo // ORSA Journal of Computing. –1995– v. 7. – n. 1. – P. 63–78.

176. Dessouky M.M. Scheduling Identical Jobs with Un-equal Ready Times on Uniform Parallel Machines to Minimize the Maximum Lateness/M.M. Dessouky// Computers & Indus-trial Engineering. – 1998 – v. 34. – n. 4. – P. 793-806.

177. Koulamas C. Scheduling on Uni-form Parallel Machines to Minimize Maximum Lateness / C. Koulamas, G. Kyparisis // Operations Research Letters. – 2000 – v. 3. – n. 3. – P. 175-179.

178. JetBrains. IntellijIdea [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/idea/.

179. Java. [Електронний ресурс] – Режим доступу до ресурсу: http://www.oracle.com/technetwork/java/index.html.

180. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: http://spring.io/.

181. Hibernate ORM [Електронний ресурс] – Режим доступу до ресурсу: http://hibernate.org/orm/.

182. Краткий обзор возможностей PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: http://postgresql.ru.net/docs/overview.html.

183. Красовский Д.В. Алгоритмы решения минимаксной задачи составления расписания / Д.В. Красовский, М.Г. Фуругян // Известия РАН. Теория и системы управления. – 2008. – №5. – С. 69–74.

ДОДАТКИ

Додаток А

Ілюстрація перестановок різних типів між паралельними пристроями

Додаток А.1

Ілюстрація перестановок різних типів між ідентичними паралельними пристроями для мінімізації максимального з виступів

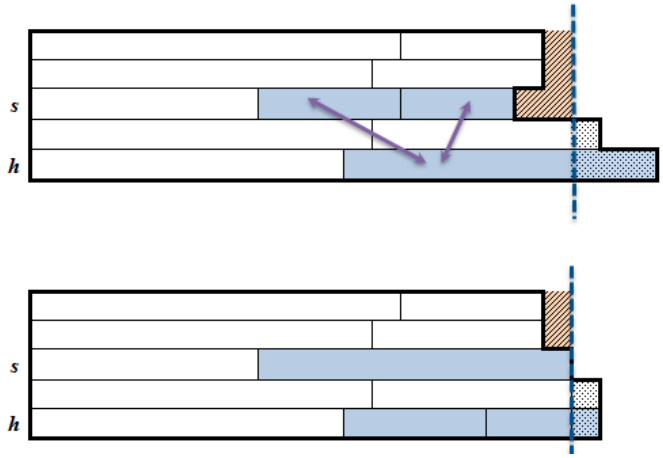


Рис. А.1.1. Перестановка типу ***1-2A***

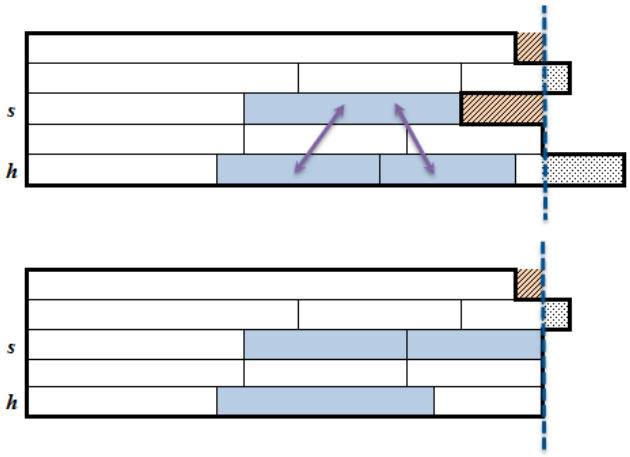


Рис. А.1.2. Перестановка типу ***2-1A***

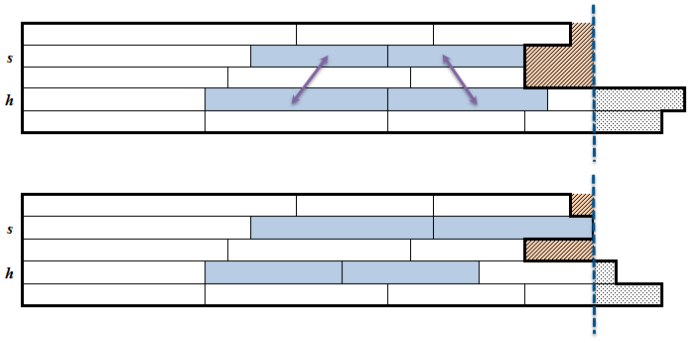


Рис. А.1.3. Перестановка типу ***2-2A***

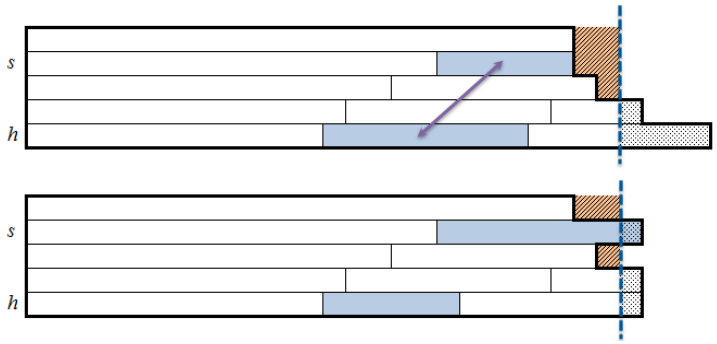


Рис. А.1.4. Перестановка типу ***1-1Б***

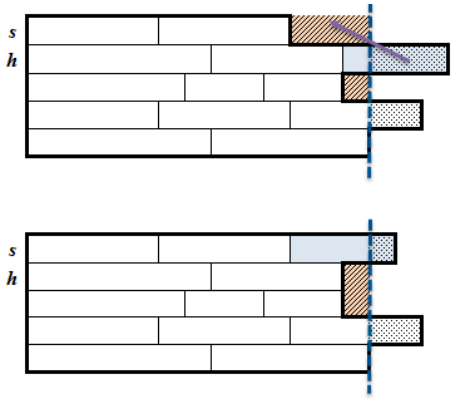


Рис. А.1.5. Перестановка типу ***0-1В***

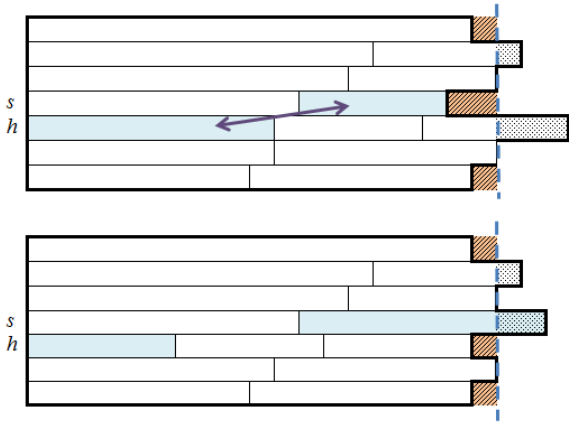


Рис. А.1.6. Перестановка типу ***1-1В***

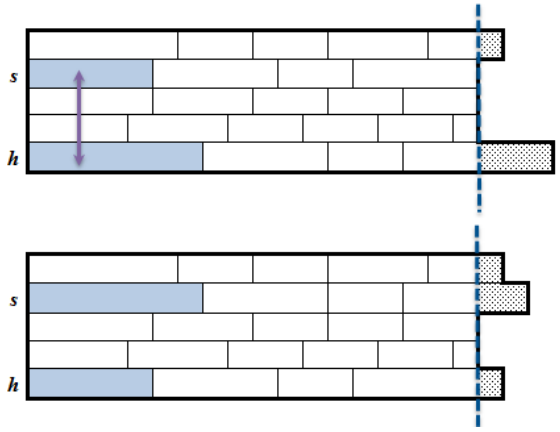


Рис. А.1.7. Перестановка типу ***1-1Г***

Додаток А.2

Ілюстрація перестановок різних типів між паралельними пристроями різної продуктивності для мінімізації максимального з виступів

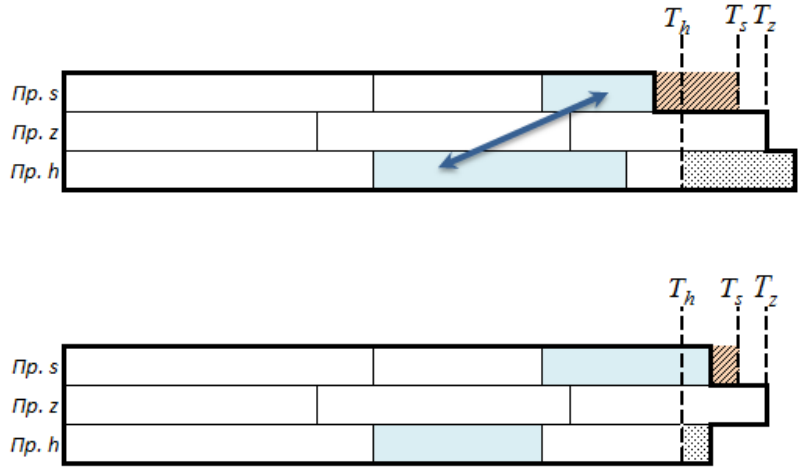


Рис. А.2.1. Приклад перестановки типу 1-1 ***А*** *k*

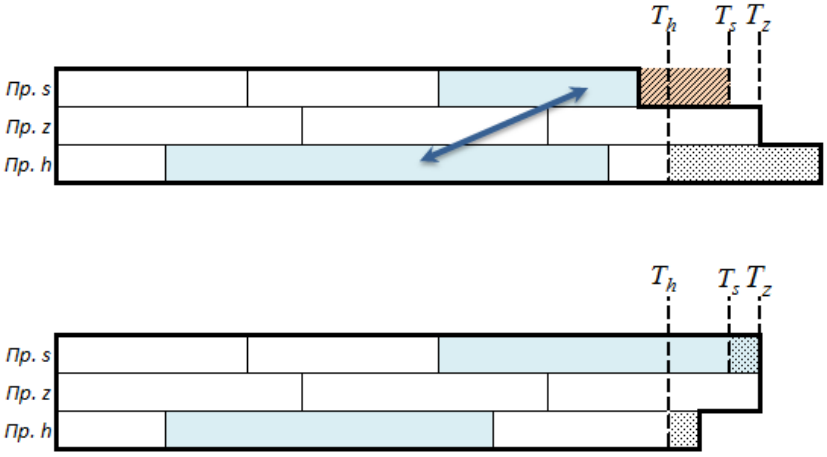


Рис. А.2.2. Приклад перестановки типу 1-1 ***Б*** *k*

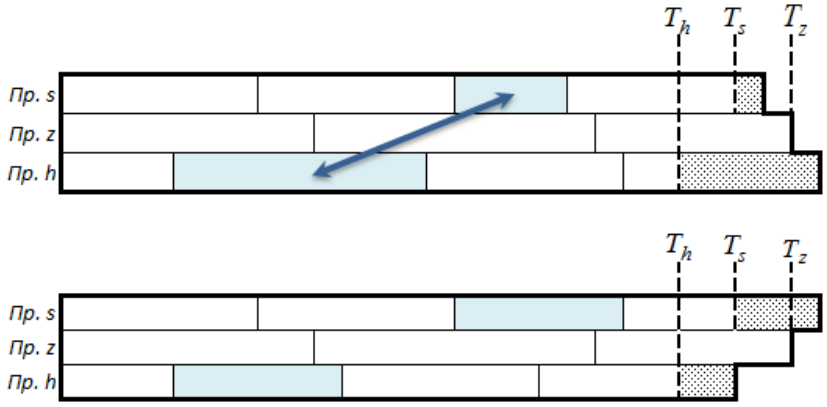


Рис. А.2.3. Приклад перестановки типу 1-1 ***Г*** *k*

Додаток А.3

Ілюстрація перестановок різних типів між ідентичними паралельними пристроями для мінімізації максимального з резервів

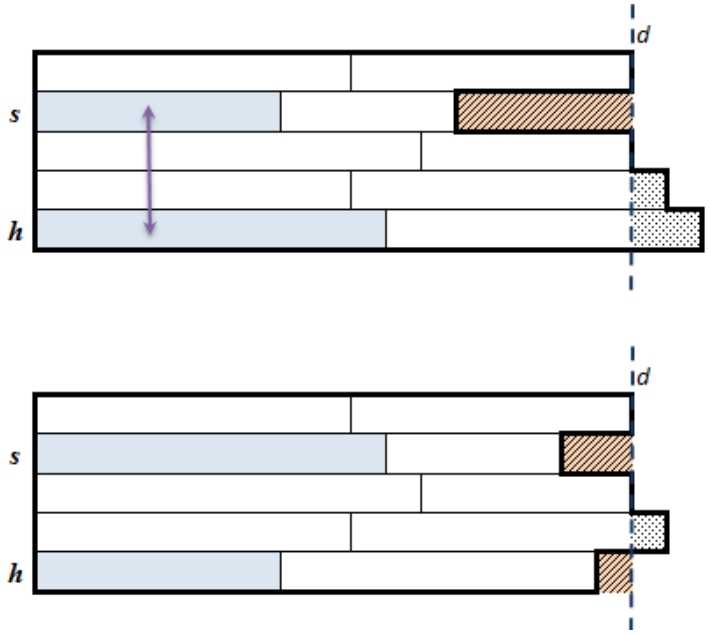


Рис. А.3.1. Приклад перестановки типу ***1-1Б*** *R*

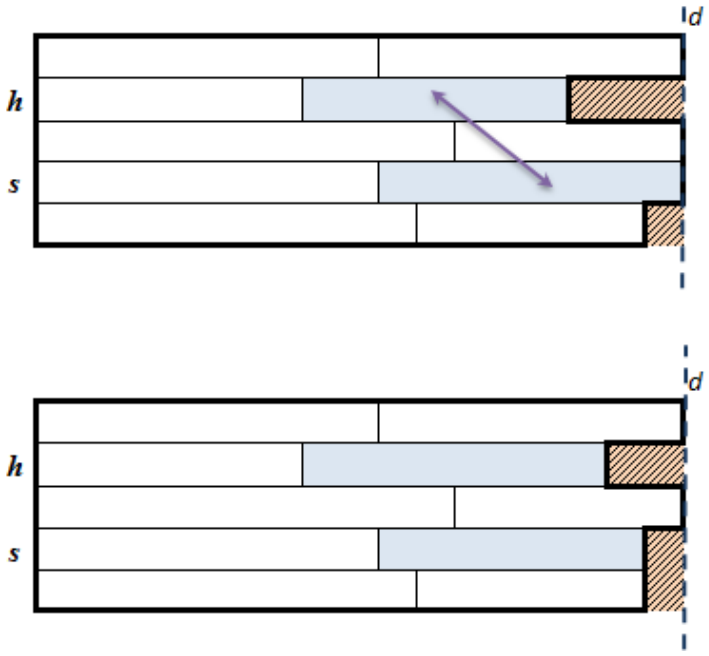


Рис. А.3.2. Приклад перестановки типу ***1-1Г*** *R*

Додаток Б

Приклад розв'язання задачі визначення максимально пізнього моменту початку виконання в допустимому розкладі завдань із спільним директивним строком паралельними пристроями різної продуктивності

Візьмемо для прикладу задачу, в якій кількість пристроїв $m=6$, коефіцієнти продуктивності яких представлені у таблиці Б.1, $n=20$ завдань, тривалості яких представлені в таблиці Б.2 (завдання перенумеровані за порядком зменшення тривалостей їх виконання), директивний строк $d=100$.

Таблиця Б.1

Коефіцієнти продуктивності пристроїв

Пристрій, i	**1**	**2**	**3**	**4**	**5**	**6**
Коеф.продукт., k_i	1	1,3	1,5	2	2,3	2,5

Таблиця Б.2

Тривалості завдань

Завдання, j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Тривалість, p_j	10	9	9	8	8	7	7	7	6	6
Завдання, j	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Тривалість, p_j	6	6	5	5	5	4	4	3	3	2

Визначимо основні розрахункові величини алгоритму.

Мінімально можливий час, за який усі пристрої могли б виконати усі завдання: $C^* = \sum_{j=1}^{n} \frac{1}{\sum_{i=1}^{m} \frac{1}{k_i p_j}} = 31\frac{47}{57}$.

«Ідеальні» зведені тривалості зайнятості пристроїв наступні:

$$c_1^* = 31\frac{47}{57},\ c_2^* = 24\frac{12}{25},\ c_3^* = 21\frac{8}{37},\ c_4^* = 15\frac{52}{57},\ c_5^* = 13\frac{41}{49},\ c_6^* = 12\frac{27}{37}.$$

Визначимо величини $\lfloor c_i^* \rfloor,\ i=1,m$: $\lfloor c_1^* \rfloor = 31$, $\lfloor c_2^* \rfloor = 24$, $\lfloor c_3^* \rfloor = 21$, $\lfloor c_4^* \rfloor = 15$, $\lfloor c_5^* \rfloor = 13$, $\lfloor c_6^* \rfloor = 12$. Далі визначимо: $\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor = 4$. Оскільки, $\delta > 0$, то для визначення контуру ідеального розкладу розв'яжемо допоміжну оптимізаційну

задачу. Визначимо e_i резерв пристрою i у неповному розкладі $\bar{\sigma}^0$: $e_1 = \frac{47}{57}$, $e_2 = \frac{12}{25}$, $e_3 = \frac{8}{37}$, $e_4 = \frac{52}{57}$, $e_5 = \frac{41}{49}$, $e_6 = \frac{27}{37}$.

Далі розподілимо δ завдань одиничної довжини між m пристроями, за умов, що пристрій i має резерв e_i, $i = \overline{1,m}$ і $\sum_{i=1}^{m} e_i = \delta$ з метою мінімізації максимального з виступів (з урахуванням продуктивності пристроїв).

Визначимо x_i кількість «одиничних» еталонних завдань, що повинні бути призначені на пристрій i, $i = \overline{1,m}$: $\sum_{i=1}^{6} x_i = 4$.

Розподілимо «одиничні» еталонні завдання за пристроями, при якому досягає мінімуму максимальний з виступів, та наведемо результати розподілу та значення цільової функції у таблиці Б.3.

Таблиця Б.3

Розподіл «одиничних» еталонних завдань за пристроями та значення цільової функції

Пристрій, i	**1**	**2**	**3**	**4**	**5**	**6**	
x_i	1	1	0	1	1	0	$\sum_{i=1}^{6} x_i = 4$
e_i	$\frac{47}{57}$	$\frac{12}{25}$	$\frac{8}{37}$	$\frac{52}{57}$	$\frac{41}{49}$	$\frac{27}{37}$	
Δ_i	$\frac{10}{57}$	$\frac{13}{25}$	0	$\frac{5}{57}$	$\frac{8}{49}$	0	$\max_{1 \le i \le 6}\{k_i(x_i - e_i)\} = \frac{13}{25}$

На рис. Б.1 наведено ілюстрацію контуру ідеального розкладу σ та на рис. Б.2 наведено контур ідеального розкладу σ' в еталонних величинах.

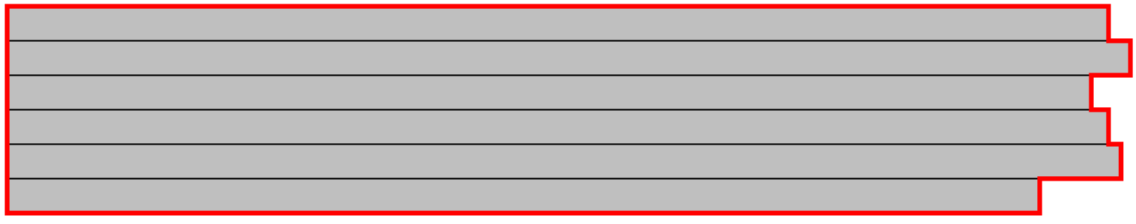


Рис. Б.1. Контур ідеального розкладу σ

Рис. Б.2. Контур ідеального розкладу $\sigma^{/}$ в еталонних величинах

Початковий розклад σ^{0}, побудований за алгоритмом ***A03***, представлено на рис. Б.3.

На рис. Б.4. представлено розклад у зведених (еталонних) тривалостях.

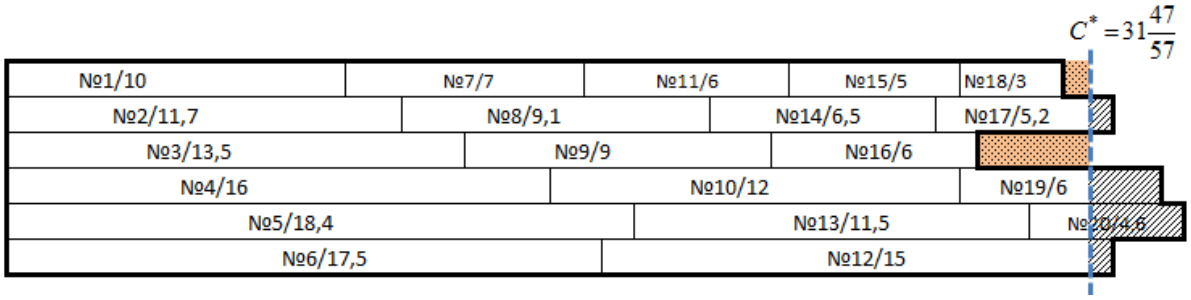


Рис. Б.3. Початковий розклад σ^{0}

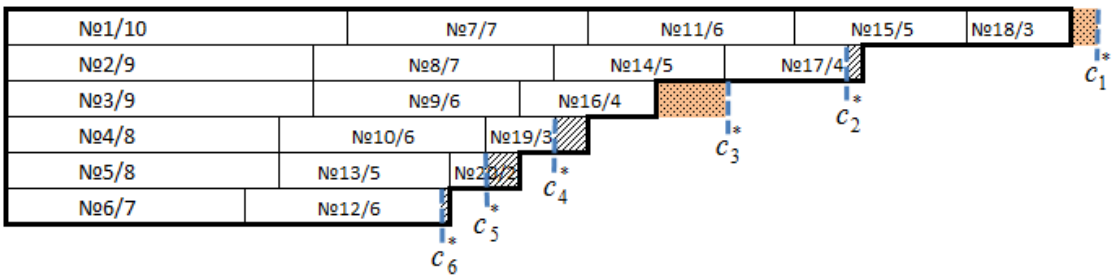


Рис. Б.4. Початковий розклад $\sigma^{0/}$ у еталонних тривалостях

Визначимо T_i^* тривалість виконання завдань в «ідеальному розкладі»:

$$T_1^* = 32,\ T_2^* = 32\frac{1}{2},\ T_3^* = 31\frac{1}{2},\ T_4^* = 32,\ T_5^* = 32\frac{1}{5},\ T_6^* = 30.$$

Розрахуємо тривалості зайнятості пристроїв у еталонних величинах:

$$T_1^{/}\left(\sigma^{0/}\right) = 32,\ T_2^{/}\left(\sigma^{0/}\right) = 25,\ T_3^{/}\left(\sigma^{0/}\right) = 21,\ T_4^{/}\left(\sigma^{0/}\right) = 16,\ T_5^{/}\left(\sigma^{0/}\right) = 14,\ T_6^{/}\left(\sigma^{0/}\right) = 12.$$

Порівняємо контури ідеального розкладу з початковим розкладом σ^{0} та $\sigma^{0/}$ (рис. Б.5 та рис. Б.6).

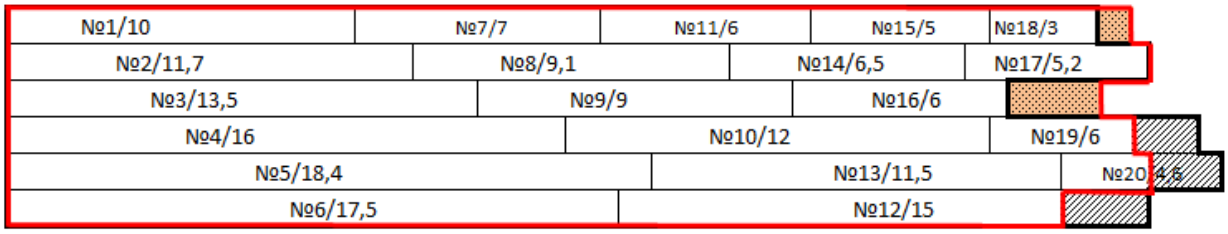


Рис. Б.5. Початковий розклад σ^0 у порівнянні з контуром ідеального розкладу

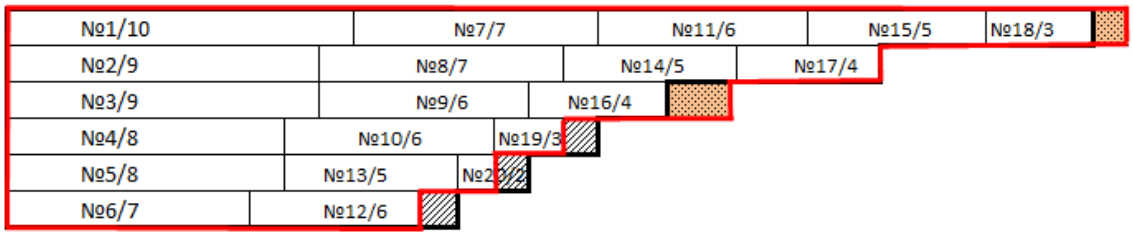


Рис. Б.6. Початковий розклад $\sigma^{0'}$ у порівнянні з контуром ідеального розкладу

Для зменшення кількості можливих варіантів при вирішенні даної задачі будемо застосовувати тільки операції обміну підтипу 1-1.

Ітерація 1

Для початкового розкладу визначимо тривалості зайнятості пристроїв:

$$T_1(\sigma^0)=31,\ T_2(\sigma^0)=32\frac{1}{2},\ T_3(\sigma^0)=28\frac{1}{2},\ T_4(\sigma^0)=34,\ T_5(\sigma^0)=34\frac{1}{2},\ T_6(\sigma^0)=32\frac{1}{2}$$

Визначимо множини $I_0(\sigma)$, $I_Z(\sigma)$ та $I_E(\sigma)$:

$$I_0(\sigma)=\{2\};$$

$$I_Z(\sigma)=\{4;5;6\}:\ Z_4(\sigma^0)=2,\ Z_5(\sigma^0)=2\frac{3}{10},\ Z_6(\sigma^0)=2\frac{1}{2};$$

$$I_E(\sigma)=\{1;3\}:\ E_1(\sigma^0)=1,\ E_3(\sigma^0)=3.$$

Визначимо пристрої $h\in I_Z(\sigma)$ та $s\in I_E(\sigma)$: $h=5$ – пристрій з максимальним значенням виступу, $s=3$ – пристрій з максимальним значенням резерву. Визначимо, яким парам завдань пристроїв 5 та 3 відповідають допустимі операції обміну (обміни, які призводять до поліпшення розкладу). Для цього спочатку виділимо ті пари завдань, для яких $\theta=\frac{p_{j_1}}{k_1}-\frac{p_{j_2}}{k_2}>0$ (до них відносяться пари завдань: 5-9, 13-16). Проаналізуємо, як співвідносяться величини θ, Z_5 та

E_3, тобто до якого із типів ***А***, ***Б*** або ***В*** відносяться потенційні операції обміну. В таблиці Б.4 представлені допустимі обміни та відповідні їх значення $\theta = \frac{p_{j_1}}{k_1} - \frac{p_{j_2}}{k_2}$ (сірим фоном відмічені недопустимі обміни, тобто обміни, для яких $\theta \le 0$ і/або не виконуються умови обміну ***А***, ***Б*** або ***В***).

Таблиця Б.4

Значення θ для допустимих обмінів завдань пристроїв 5-3

				Завдання пристрою 3		
				$k_3 = 1{,}5$		
				№3	№9	№16
				13,5	***9***	***6***
Завдання пристрою 5	$k_5 = 2{,}3$	№5	***18,4***		2	
		№13	***11,5***			1
		№20	***4,6***			

В таблиці Б.5 наведені параметри допустимих для пари пристроїв 5-3 обмінів.

Таблиця Б.5

Параметри допустимих обмінів для пари пристроїв 5-3

Пара завдань	Z_5	E_3	θ	$\frac{Z_5}{k_5} + \frac{E_3}{k_3} - \theta$	**Величина покращення**	**Тип обміну**
5-9	$2\frac{3}{10}$	3	2	1	1	***Бk***
13-16	$2\frac{3}{10}$	3	1	2	1	***Аk***

Наслідки обміну між парою завдань 13-16 кращі, оберемо цей обмін та виконаємо його. На рис. Б.7 зображено розклад σ^1, отриманий в результаті обміну.

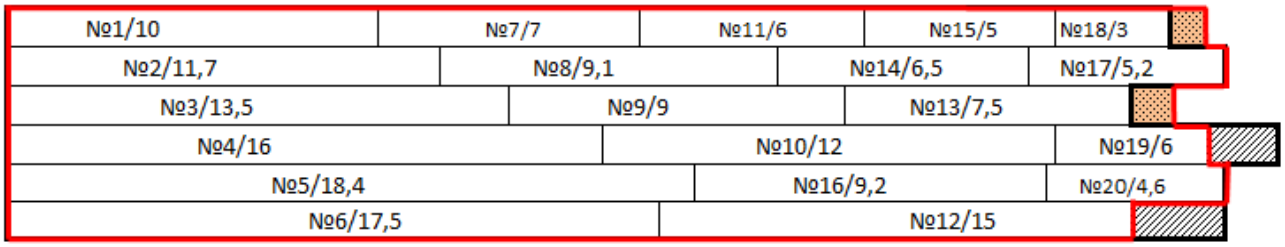


Рис. Б.7. Розклад σ^1

Для розкладу σ^1 не виконується ДУО 2, продовжуємо роботу алгоритму.

Ітерація №2

Для поточного розкладу σ^1 маємо: $T_1(\sigma^1)=31$, $T_2(\sigma^1)=32\frac{1}{2}$, $T_3(\sigma^1)=30$, $T_4(\sigma^1)=34$, $T_5(\sigma^1)=32\frac{1}{5}$, $T_6(\sigma^1)=32\frac{1}{2}$.

Визначимо $I_0(\sigma)$, $I_Z(\sigma)$ та $I_E(\sigma)$:

$I_0(\sigma)=\{2;5\}$;

$I_Z(\sigma)=\{4;6\}$: $Z_4(\sigma^0)=2$, $Z_6(\sigma^0)=2\frac{1}{2}$;

$I_E(\sigma)=\{1;3\}$: $E_1(\sigma^0)=1$, $E_3(\sigma^0)=1\frac{1}{2}$.

Визначимо пристрої $h\in I_Z(\sigma)$ та $s\in I_E(\sigma)$: $h=6$ – пристрій з максимальним значенням виступу, $s=3$ – пристрій з максимальним значенням резерву. В таблиці Б.6 представлені допустимі операції обміну для вибраної пари пристроїв та відповідні їм значення θ, а в таблиці Б.7 наведені параметри цих обмінів.

Таблиця Б.6

Значення θ для допустимих обмінів завдань пристроїв 6-3

				Завдання пристрою 3		
				$k_3=1{,}5$		
				№3	№9	№13
				13,5	***9***	***7,5***
Завдання пристрою 6	$k_6=2{,}5$	№6	***17,5***		1	
		№12	***15***			1

Таблиця Б.7

Параметри допустимих обмінів для пари пристроїв 6-3

Пара завдань	Z_6	E_3	θ	$\frac{Z_6}{k_6}+\frac{E_3}{k_3}-\theta$	**Величина покращення**	**Тип обміну**
6-9	$2\frac{1}{2}$	$1\frac{1}{2}$	1	1	1	***A**k*
12-13	$2\frac{1}{2}$	$1\frac{1}{2}$	1	1	1	***A**k*

Наслідки усіх операцій обміну однакові, оберемо обмін між завданнями 6-9 (перший в списку) та виконаємо його. На рис. Б.8 зображено розклад σ^2, отриманий в результаті цього обміну.

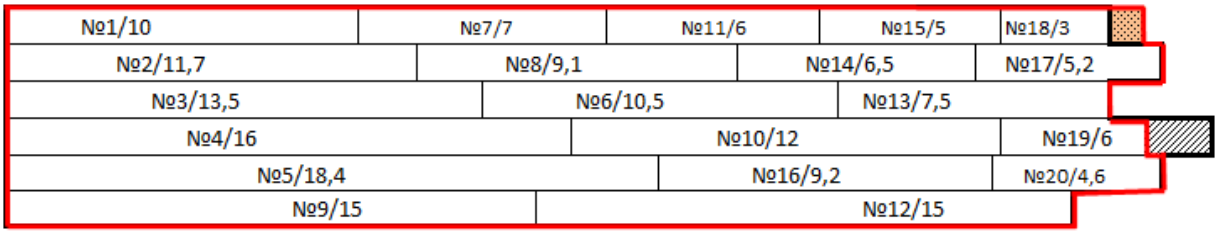


Рис. Б.8. Розклад σ^2

Для розкладу σ^2 не виконується достатня умова оптимальності 2, продовжуємо роботу алгоритму.

Ітерація №3

Для поточного розкладу σ^2 маємо: $T_1(\sigma^1)=31$, $T_2(\sigma^1)=32\frac{1}{2}$, $T_3(\sigma^1)=31\frac{1}{2}$, $T_4(\sigma^1)=34$, $T_5(\sigma^1)=32\frac{1}{5}$, $T_6(\sigma^1)=30$.

Визначимо множини $I_0(\sigma)$, $I_Z(\sigma)$ та $I_E(\sigma)$:

$I_0(\sigma)=\{2;3;5;6\}$;

$I_Z(\sigma)=\{4\}$: $Z_4(\sigma^0)=2$;

$I_E(\sigma)=\{1\}$: $E_1(\sigma^0)=1$.

Визначимо пристрої $h\in I_Z(\sigma)$ та $s\in I_E(\sigma)$: $h=4$ – пристрій з максимальним значенням виступу, $s=1$ – пристрій з максимальним значенням резерву. В таблиці Б.8 представлені допустимі операції обміну для вибраної пари пристроїв та відповідні їм значення θ, а в таблиці Б.9 наведені параметри цих обмінів.

Таблиця Б.8

Значення θ для допустимих обмінів завдань між пристроями 4-1

				Завдання пристрою 1				
				$k_1 = 1$				
				№1	№7	№11	№15	№18
				10	***7***	***6***	***5***	***3***
Завдання пристрою 4	$k_4 = 2$	№4	***16***		1			
		№10	***12***				1	
		№19	***6***					

Таблиця Б.9

Наслідки можливих обмінів для завдань пристроїв 4-1

Пара завдань	Z_4	E_1	θ	$\frac{Z_4}{k_4} + \frac{E_1}{k_1} - \theta$	**Величина покращення**	**Тип обміну**
4-7	2	2	1	2	1	***A**k*
10-15	2	2	1	1	1	***A**k*

Наслідки усіх операцій обміну однакові, оберемо обмін між завданнями 4-7 (перший в списку) та виконаємо його. На рис. Б.9 зображено розклад σ^3, отриманий в результаті цього обміну.

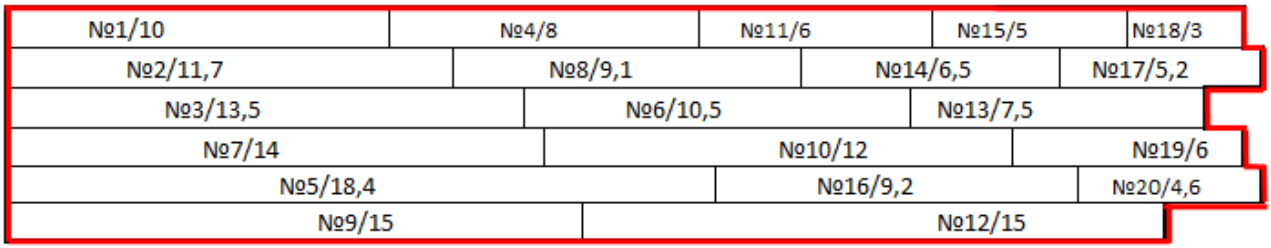


Рис. Б.9. Розклад σ^3

Отриманий розклад є оптимальним за другою достатньою умовою оптимальності. Визначимо для нього максимально пізній момент запуску, при якому всі завдання будуть виконані до директивного строку:

$$r_{\max} = r(\sigma) = d - \max_i T_i(\sigma) = 100 - 32{,}5 = 67{,}5\,.$$

Додаток В

Результати проведення експериметів

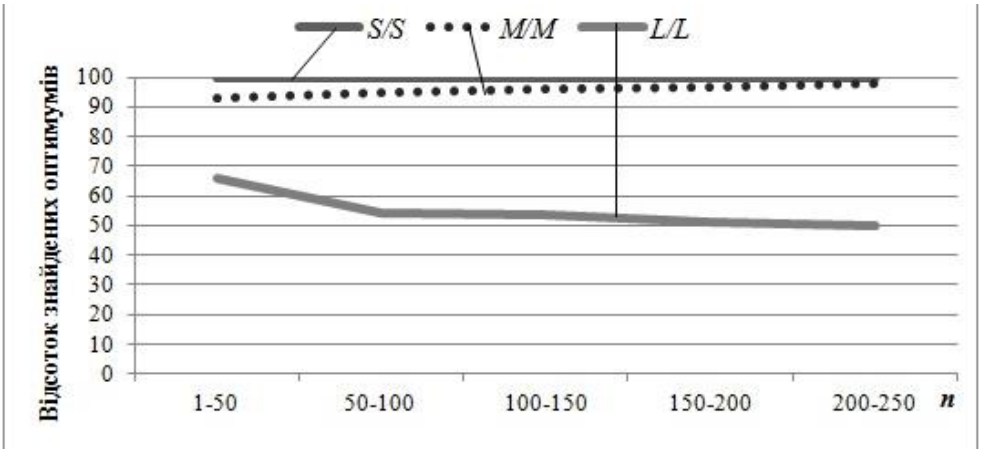


Рис. В.1. Залежність точності алгоритму від кількості завдань n

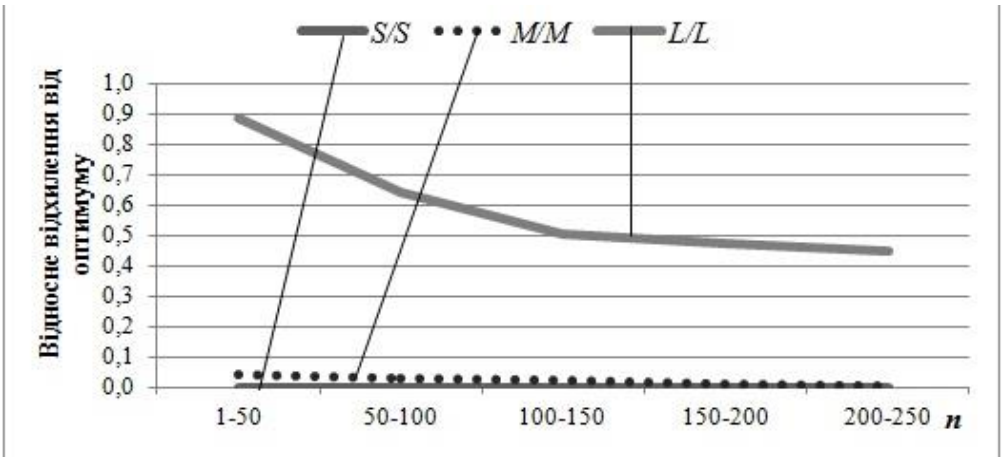


Рис. В.2. Залежність ступеня досягнення оптимуму від кількості завдань n

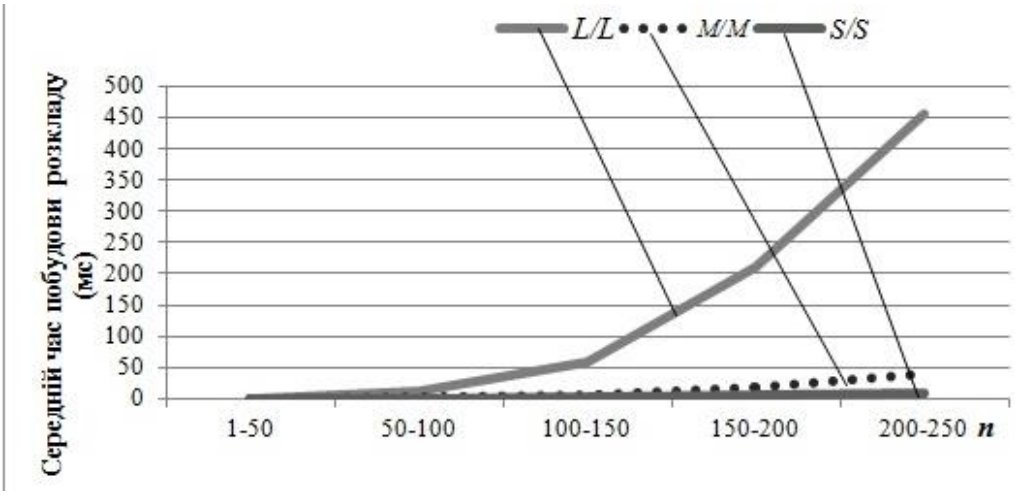


Рис. В.3. Залежність часу роботи (мс) алгоритму від кількості завдань n

Додаток Г

Опис класів інструментального комплексу та їх основних функцій

Додаток Г.1

Класи інструментального комплексу

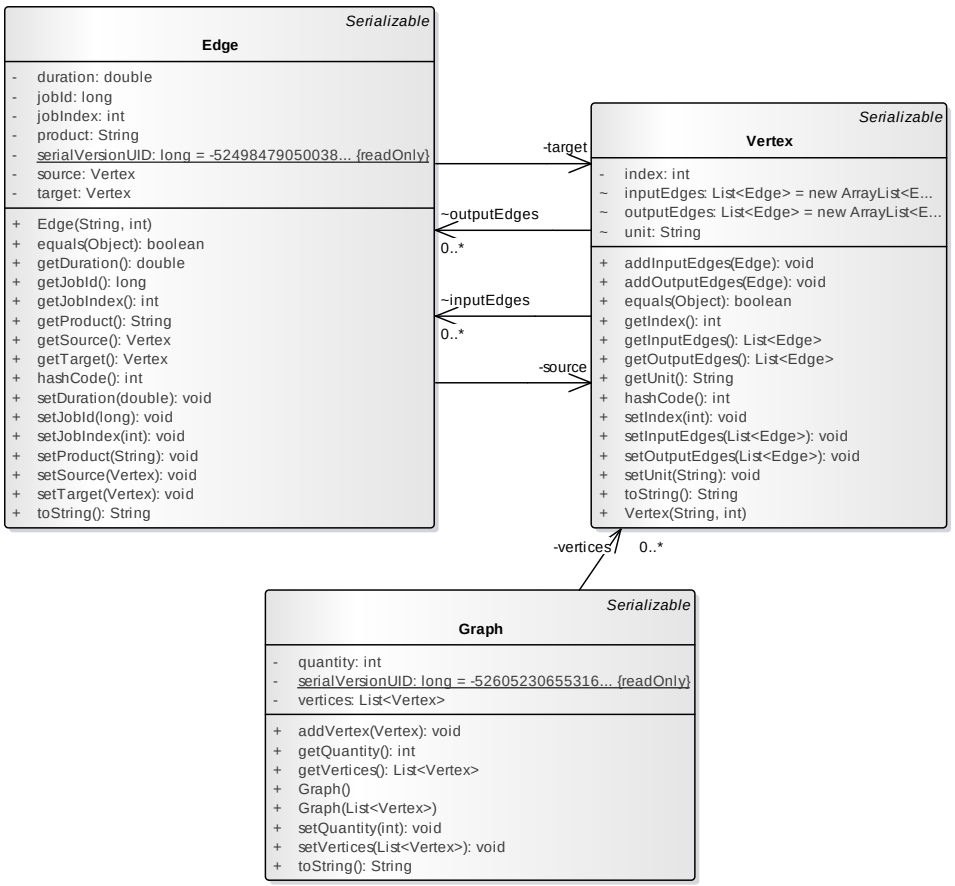


Рис. Г.1.1. Діаграма класів пакету com.algorithm.entities

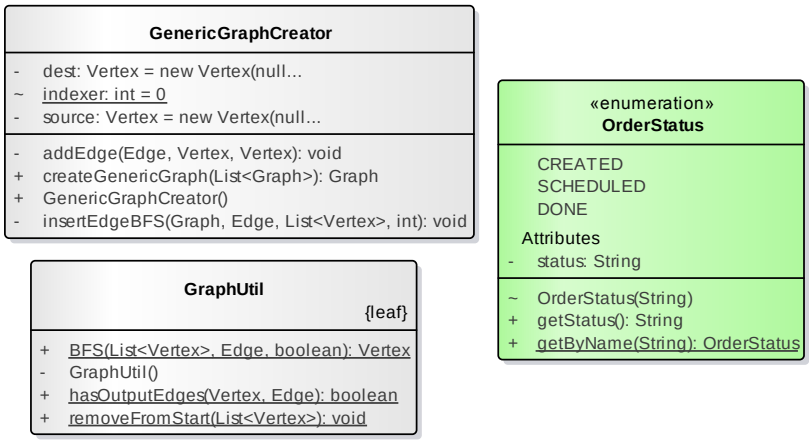


Рис. Г.1.2. Діаграма класів пакету com.algorithm.scheduling

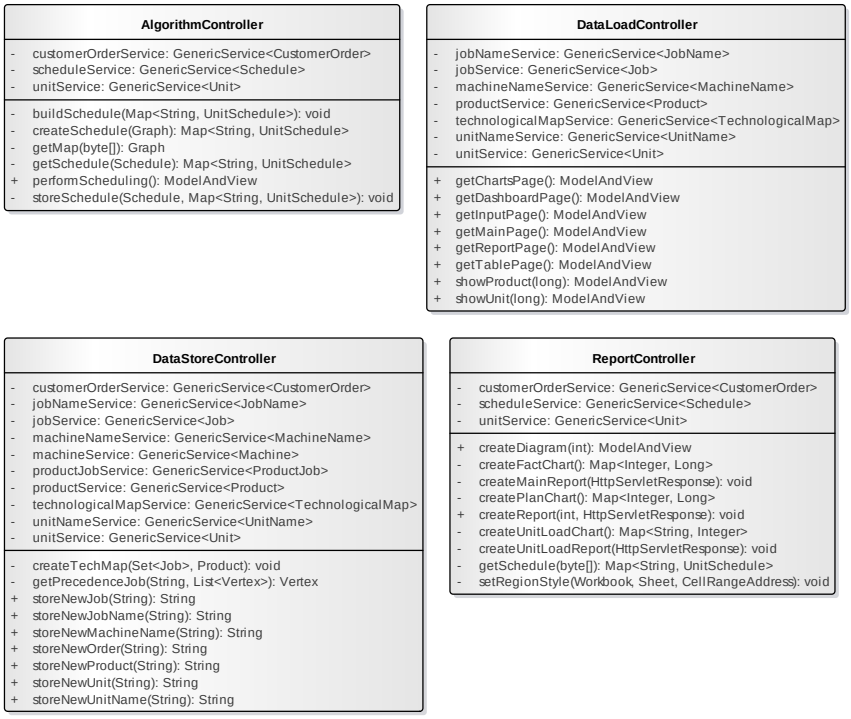


Рис. Г.1.3. Діаграма класів пакету com.controllers

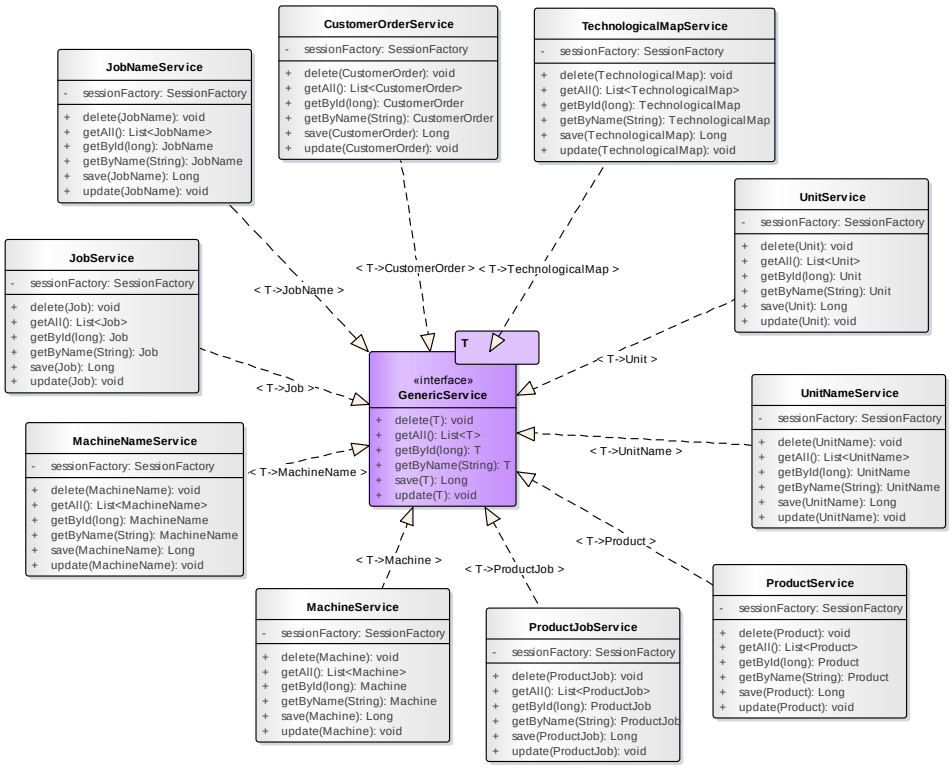


Рис. Г.1.4. Діаграма класів пакету com.services

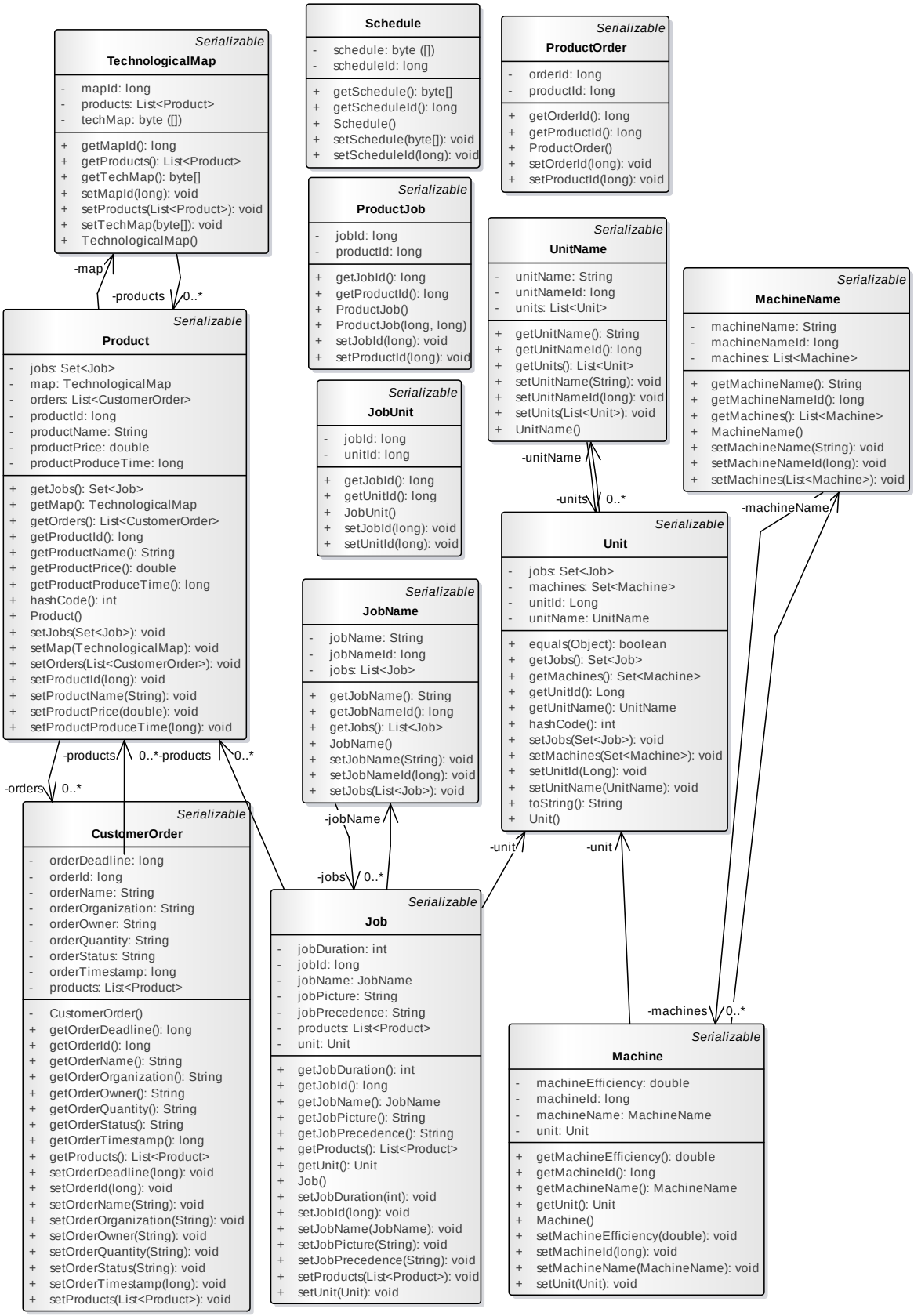


Рис. Г.1.5. Діаграма класів пакету com.sql.entities

Додаток Г.2

Основні функції класів

Таблиця Г.2.1

Опис основних функцій

Клас	Функція	Опис функції
Entities<T> відображає відповідний запис таблиці бази даних у об’єкт	`public T getField()`	Повертає значення відповідного поля
	`public void setField(T t)`	Встановлює значення відповідного поля
GenericService інтерфейс, який оголошує основні операції над даними бази даних	`List<T> getAll()`	Дістає всі записи відповідної таблиці з бази даних
	`T getById(long id)`	Дістає запис відповідної таблиці з бази даних по відповідному ідентифікатору
	`T getByName(String name)`	Дістає запис відповідної таблиці з бази даних по імені
	`Long save(T t)`	Зберігає новий запис до бази даних та повертає його унікальний ідентифікатор
	`void update(T t)`	Оновлює існуючий запис в базі даних
	`void delete(T t)`	Видаляє з бази даних відповідний запис
AlgorithmController реалізує основні алгоритми	`public ModelAndView performScheduling()`	Обробляє запит на складання розкладу
	`public Graph getMap(byte map)`	Здійснює конвертацію бінарних даних в об’єкт
	`private Map<String, UnitSchedule> createSchedule(Graph genericGraph)`	Створює розклад
	`public void buildSchedule(Map<String, UnitSchedule > scheduleMap)`	Розподіляє завдання по пристроях

Продовження табл. Г.2.1

Клас	Функція	Опис функції
DataLoadController відповідає за завантаження основних сторінок та даних	*public ModelAndView getMainPage()*	Завантажує головну сторінку
	public ModelAndView getTablePage()	Завантажує сторінку з усіма введеними даними
	public ModelAndView getChartsPAge()	Завантажує сторінку з діаграмами
	public ModelAndView getDashboardPage()	Завантажує сторінку для введення замовлень
	public ModelAndView showProduct(@RequestParam(name = "id"))	Завантажує сторінку з деталями продукту
DataStoreController обробляє запити на додавання нових даних	*public @ResponseBody String storeNewProduct(@RequestBody String productJson)*	Додає новий виріб
	public @ResponseBody String storeNewUnit(@RequestBody String unitJson)	Додає нову дільницю
	public @ResponseBody String storeNewJob(@RequestBody String jobJson)	Додає нове завдання
	public @ResponseBody String storeNewMachineName(@RequestBody String machineJson)	Додає нове ім'я пристрою
	public @ResponseBody String storeNewUnitName(@RequestBody String unitJson)	Додає нове ім'я виробничої дільниці
	public @ResponseBody String storeNewJobName(@RequestBody String jobJson)	Додає нове ім'я завдання
	public @ResponseBody String storeNewOrder(@RequestBody String orderJson)	Додає нове замовлення
	private void createTechMap(List<Job> jobs, Product product)	Створює технологічну карту виробу
	private Vertex getPrecedenceJob(String index, List<Vertex> vertices)	Повертає передуючу вершину

Продовження табл. Г.2.1

Клас	Функція	Опис функції
GenericGraphCreator реалізує базовий фунціонал для створення узагальненої технологічної карти	`public Graph createGenericGraph (List<Graph> vertices)`	Створює узагальнений граф
	`private void insertEdgeBFS(Graph genericGraph, Edge e, List<Vertex> startVertices, int quantity)`	Додає ребро до узагальненого графу
	`private void addEdge(Edge e, Vertex source, Vertex target)`	Додає вершинам відповідне ребро у список ребер
GraphUtil реалізує базові операції над графами	`public static boolean hasOutputEdges(Vertex v, Edge e)`	Перевіряє чи є у вершини віхідні ребра
	`public static removeFromStart(List<Vertex> startVertices)`	Видаляє вершину з списку початкових вершин
	`public static Vertex BFS(List<Vertex> start, Edge e, boolean isSource)`	Реалізація пошуку в ширину
Edge відображає ребро графа	`public T getField()`	Повертає значення відповідного поля
	`public void setField(T t)`	Встановлює значення відповідного поля
Graph відображає граф, містить в собі список першин	`public T getField()`	Повертає значення відповідного поля
	`public void setField(T t)`	Встановлює значення відповідного поля
	`public void addVertex(Vertex v)`	Додає вершину до списку вершин
Vertex відображає вершину графа	`public T getField()`	Повертає значення відповідного поля
	`public void setField(T t)`	Встановлює значення відповідного поля
	`public void addInputEdge(Edge e)`	Додає ребро до списку вхідних ребер
	`public void addOutputEdge(Edge e)`	Додає ребро до списку вих. ребер
	`public T getField()`	Повертає значення відповідного поля
	`public void setField(T t)`	Встановлює знач. відповідного поля
MachineSchedule містить завдання, які повинні виконуватись на пристрої	`public void addJob(ScheduleJob job)`	Додає завдання до списку
	`public T getField()`	Повертає значення відповідного поля
	`public void setField(T t)`	Встановлює значення відповідного поля

Продовження табл. Г.2.1

Клас	Функція	Опис функції
ScheduleJob містить інформацію про завдання	`public T getField()`	Повертає значення відповідного поля
	`public void setField(T t)`	Встановлює значення відповідного поля
UnitSchedule містить розклад для виробничої дільниці	`public void genericJob(double job)`	Додає завдання до списку загального списку завдань
	`public void addJob(double job mint machine)`	Додає завдання до конкретного пристрою
	`public int getMachineMinReleaseIndex()`	Повертає індекс пристрою з найменшим часом звільнення

Додаток Д

Приклад побудови узагальненоготехнологічного графу

Нехай обслуговуюча система складається з трьох виробничих вузлів (див. рис. Д.1). Необхідно виготовити множину з п’яти виробів p_i, $i=\overline{1,5}$, графи відношень безпосереднього передування завдань яких із відповідними проміжними графами G_i^U представлені на рис. Д.2–Д.5 (тут завдання позначаються парою індексів «індекс виробу i: індекс завдання j в межах технологічної карти цього виробу», що розділені двокрапкою).

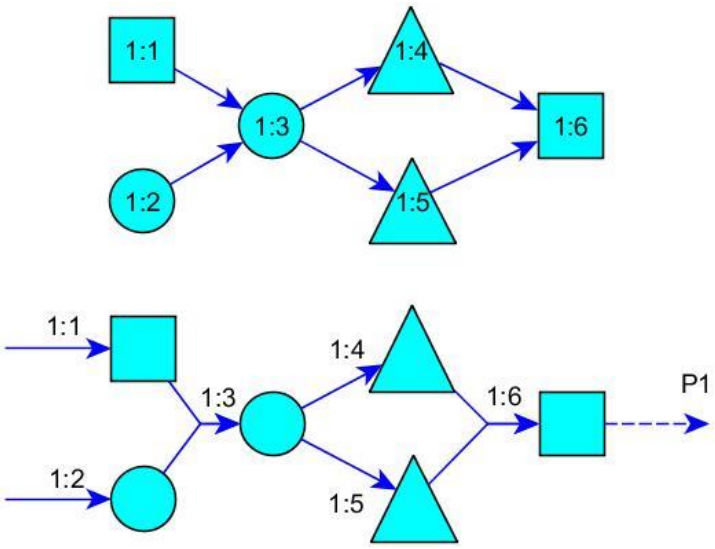


Рис. Д.1. Графи G_1 (зверху) та G_1^U (знизу)

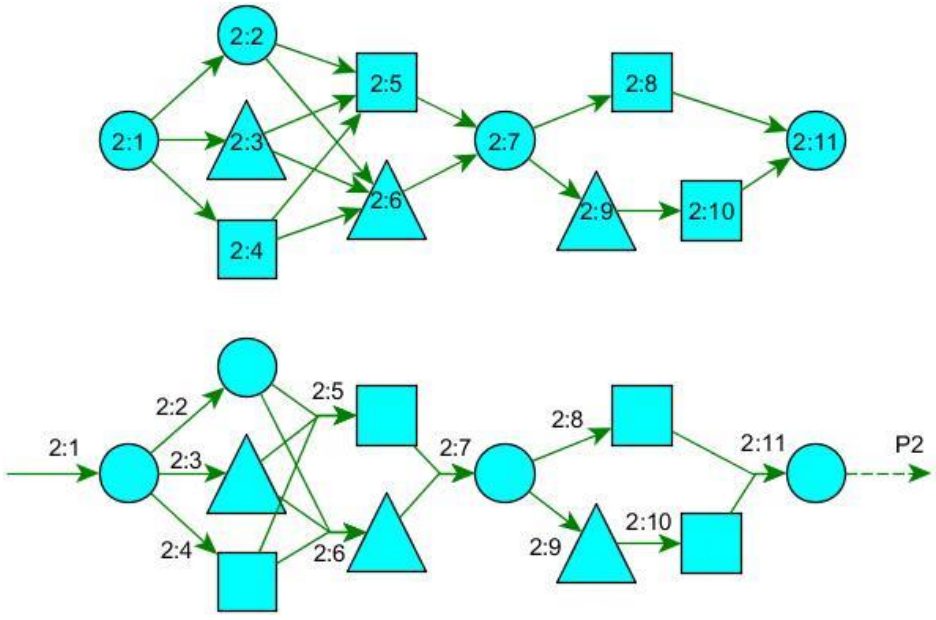


Рис. Д.2. Графи G_2 (зверху) та G_2^U (знизу)

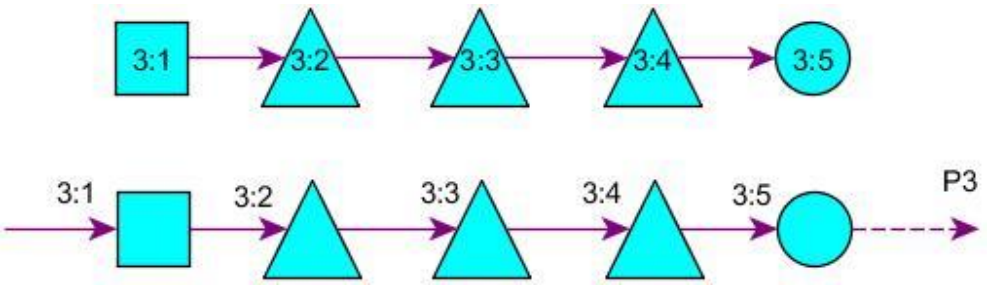


Рис. Д.3. Графи G_3 (зверху) та G_3^U (знизу)

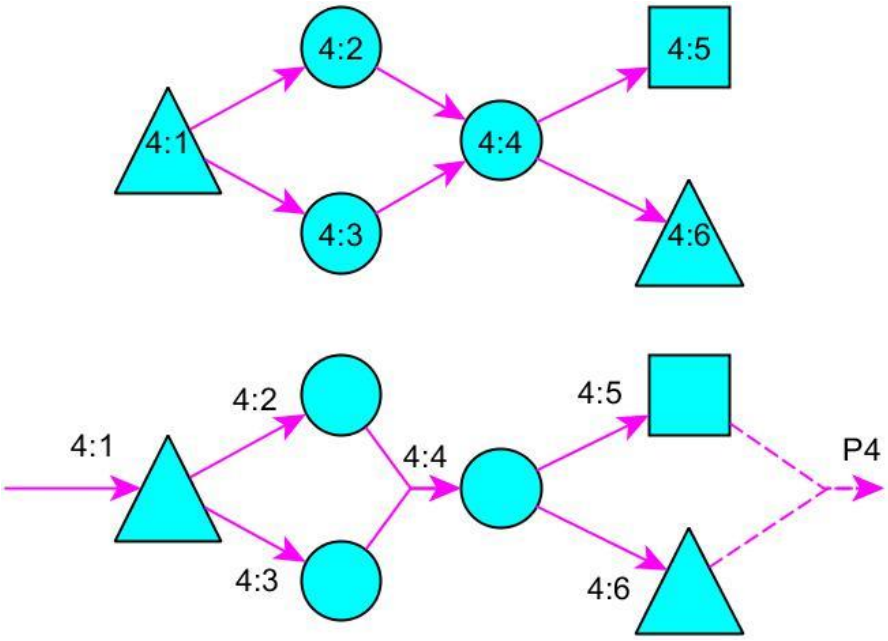


Рис. Д.4. Графи G_4 (зверху) та G_4^U (знизу)

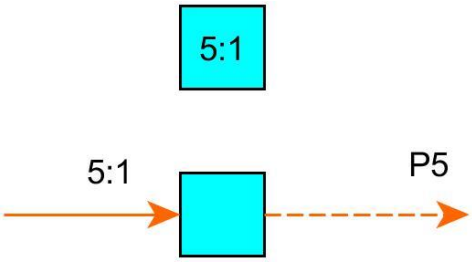


Рис. Д.5. Графи G_5 (зверху) та G_5^U (знизу)

Узагальнений граф для представленої множини виробів наведено на рис. Д.6.

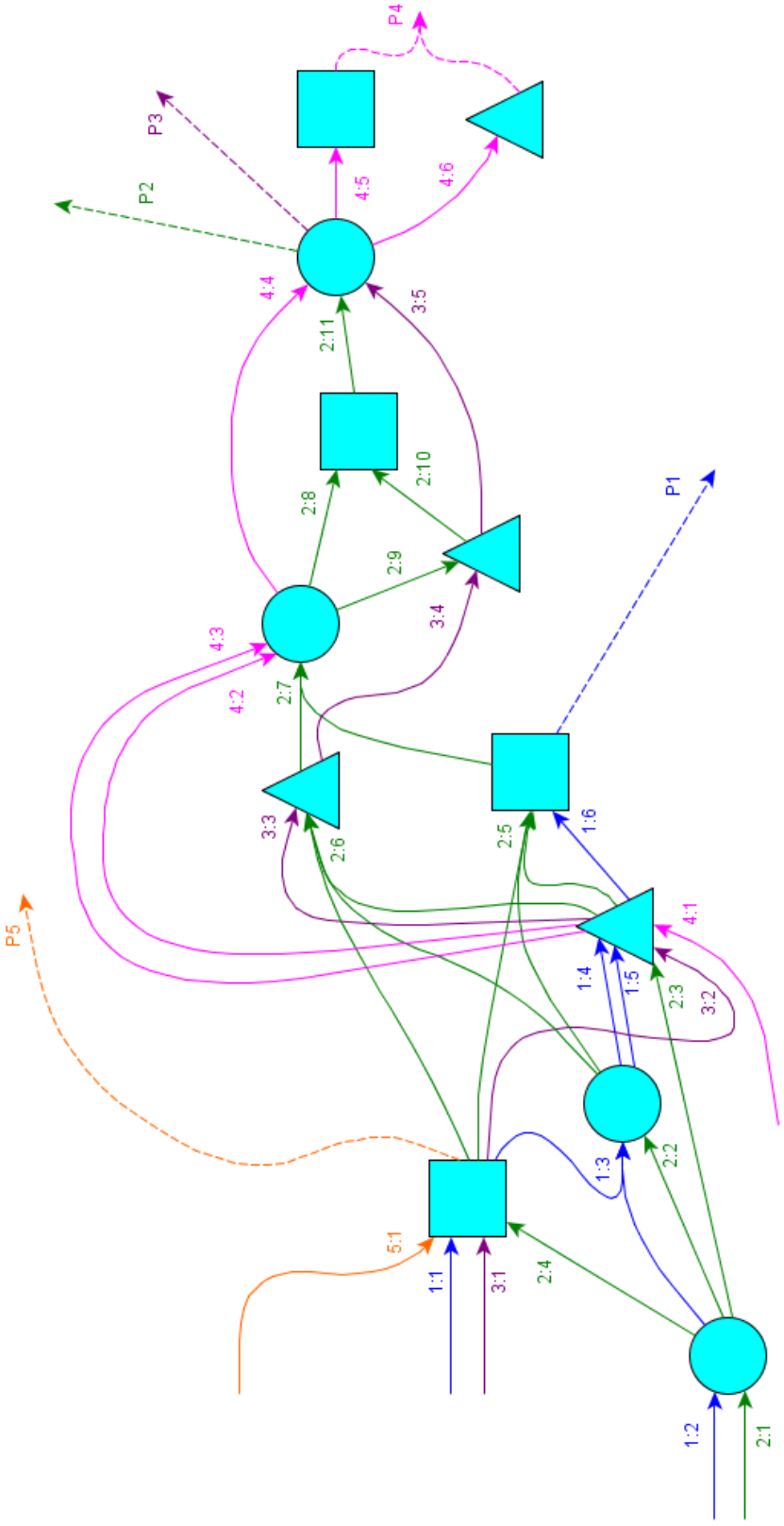


Рис. Д.6. Узагальнений граф передування завдань G для прикладу 4.2

Додаток Ж

Акти впровадження

«ЗАТВЕРДЖУЮ»

Директор
ТОВ "БІ ДЖІ ЕФ ЦЕНТРАЛ ЄУРОП"
_____ О.Г. Березовський
«__» березня 2015 р.

АКТ

використання результатів дисертаційної роботи
аспірантки НТУУ «КПІ» Сперкач Майї Олегівни
у науково-технічних розробках ТОВ "БІ ДЖІ ЕФ ЦЕНТРАЛ ЄУРОП"

Цим актом підтверджується впровадження результатів дисертаційних досліджень аспірантки НТУУ «КПІ» Сперкач Майї Олегівни у науково-технічних розробках ТОВ "БІ ДЖІ ЕФ ЦЕНТРАЛ ЄУРОП", в яких використовуються розроблені моделі, методи та підходи до оперативно-календарного планування виробництва при розробці програмного забезпечення для управління проектами.

Цей документ не є підставою для фінансових розрахунків.

Заст. директора Д.В. Савченко

АРТЕМ
ДЕРЖАВНА АКЦІОНЕРНА
ХОЛДИНГОВА КОМПАНІЯ

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО
“ВИРОБНИЧО-КОМЕРЦІЙНА ФІРМА “АС”

Україна, 04050
м. Київ, вул .Мельникова, 2/10

22.07.2015 р. № 261

АКТ

використання результатів дисертаційної роботи
аспірантки НТУУ «КПІ» Сперкач Майї Олегівни
у науково-технічних розробках
Приватного акціонерного товариства «Виробничо-комерційна фірма «АС»

Цим актом підтверджується впровадження результатів дисертаційних досліджень аспірантки НТУУ «КПІ» Сперкач Майї Олегівни у виробничих розробках Приватного акціонерного товариства «Виробничо-комерційна фірма «АС», а саме методів й моделей інформаційної технології при розробці програмного забезпечення для багаторівневого планування виробництва

Цей документ не є підставою для фінансових розрахунків.

Директор Климчук О.М.

Україна * м. Київ * Приватне акціонерне товариство «ВИРОБНИЧО-КОМЕРЦІЙНА ФІРМА «АС» Ідентифікаційний код 19326153

«ЗАТВЕРДЖУЮ»
Декан ФІОТ НТУУ «КПІ»
О.А. Павлов
«01» 12 2015 р.

АКТ
використання результатів дисертаційної роботи
аспірантки НТУУ «КПІ» Сперкач Майї Олегівни
у навчальному процесі кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут»

Цим актом підтверджується впровадження результатів дисертаційних досліджень аспірантки НТУУ «КПІ» Сперкач Майї Олегівни у навчальному процесі кафедри автоматизованих систем обробки інформації та управління НТУУ «КПІ», а саме у викладанні курсу «Теорії розкладів».

Цей документ не є підставою для фінансових розрахунків.

В.о.зав.каф. О.Г. Жданова